On using Deep Reinforcement Learning for Multi-Domain SFC placement

Nassima Toumi[1], Miloud Bagaa[2], Adlen Ksentini[1]
[1] EURECOM, Sophia-Antipolis, France
[2] CSC-IT Center for Science Ltd., Espoo, Finland
Email: {nassima.toumi, adlen.ksentini}@eurecom.fr, miloud.bagaa@csc.fi

***Abstract*—Service Function Chaining (SFC) has emerged as a promising technology for 5G and beyond. It leverages Network Function Virtualization (NFV) and Software Defined Networking (SDN) and allows the decomposition of a given service into a set of blocks that successively process data. The SFC placement issue has been extensively studied in the literature, and different solutions have been proposed using mathematical models and heuristics. More recently, Reinforcement Learning (RL) has emerged as a tool for decision-making that allows agents to elaborate policies based on the environment's feedback. In this paper, we study the benefits of using Deep Reinforcement Learning methods for the multi-domain SFC placement problem. We propose a Deep Deterministic Policy Gradient (DDPG) approach, where Linear Physical Programming is employed to generate rewards that reflect the solution's quality in terms of cost and latency. Through our experiments, we are able to demonstrate the efficiency of our approach with results that satisfy the SLA requirements.**

I. Introduction

Service Function Chaining (SFC) refers to the process of steering traffic through a set of functions in an ordered manner to deliver an end-to-end service [1]. Multiple works in the literature have studied SFC under different aspects: placement and chaining, scheduling and orchestration, and life-cycle management. However, most of these works don't account for the multi-domain scenario, which is when the components of a Service are required to be deployed on different domains [2], then chained together to compose an SFC that spans multiple domains. For security reasons, the domains are reluctant to disclose details on their infrastructure [3], which makes the optimal placement of multi-domain SFCs more difficult due to incomplete information on the global network topology.

On the other hand, Machine Learning (ML) has gained popularity as a result of recent improvements in computational capabilities. Indeed, with the help of sufficient data, models can be trained to solve multiple problems efficiently. ML can be applied to the operations and management of networks in several ways: traffic prediction, classification and routing, and also congestion control, resource management, and fault management [4]. One particular field of ML, called Reinforcement Learning (RL), has gained a lot of attention in recent years as a tool for automated decision-making [5]. With Reinforcement Learning, the agent relies on the rewards from past experience and the environment's feedback to build a decision policy that would be continuously improved at run-time and that can adapt to changes in the environment. Compared to the classic optimization algorithms, a Reinforcement Learning model that has been sufficiently trained can provide solutions in near real-time, regardless of the size of the problem.

In this paper, we apply RL for multi-domain SFC placement. We propose a model that constructs SFC placement policies based on the environment's feedback while taking into account the limited visibility of the local infrastructure. Notice that in this contribution, a single domain is defined as a network infrastructure that is autonomous in terms of orchestration policies; consequently, independent divisions of the same administrative entity can also be considered as separate domains. The main contributions of this paper are two-fold: First, we propose a multi-domain SFC placement framework that includes a Deep Reinforcement Learning agent. Second, we train that agent to perform the multi-domain SFC placement on Partially Observable states using rewards that express the quality of the obtained placements using Linear Physical Programming. The remainder of this paper is organized as follows. We first provide a brief background on the existing contributions to the topic in section II. Then we introduce our proposed multi-domain framework and its components in section III. Next, we detail our model's states, actions, and rewards in section IV. Afterward, we describe our evaluation testbed and discuss the obtained results in section V before concluding the paper in section VI.

II. Background

The multi-domain setting takes place when the SFC is split between multiple administrative domains. As a result, the SFC request is partitioned into multiple sub-chains that are deployed on each selected domain, which causes additional complexity. Indeed, in the multi-domain context, the different domain operators don't disclose the complete information on their infrastructure, making the SFC placement process more difficult due to insufficient information [3].

The multi-domain SFC placement issue has been tackled by a number of works, where two main approaches have been employed: a distributed one, where the domains exchange messages following a pre-determined protocol until the convergence to an optimal value [6], and a hierarchical approach, where a logically centralized entity is entrusted by the domains to partial information on their infrastructure and performs a preliminary placement and partitioning of the SFC request such as in [7]. Multiple mathematical models and algorithms have been used to optimize the placement, such as Integer Linear Programming, different heuristics, Game Theory, or Physical Programming [3], [8]. Although these proposals provide near-optimal placement solutions, the processing time increases along with the size of the problem and might reach values that wouldn't be acceptable in real-time deployments. To cope with this issue, recent works have turned to use emerging methods such as Reinforcement Learning, which presents the advantage of providing solutions in near real-time once the model has been sufficiently trained. This method also doesn't require large amounts of data for training, as the agent constructs and adjusts its decision policy using rewards and penalties from the environment. Reinforcement Learning has been employed for dynamic adaptive SFC placement, as well as traffic prediction, and proactive resource allocation [9]–[12].

The work in [13] proposed a solution for Network Slice Deployment on multiple substrate networks based on a Deep Reinforcement Learning (DRL) model, but supposed that the agent had access to complete information on the domains' infrastructure. Similarly, Swapna *et al.* [14] provide a solution for Slice resource orchestration on multiple domains based on DRL, but it also ignores the limited visibility aspect. The authors in [15] detail a privacy-preserving solution for Virtual Network Embedding (VNE) on multiple domains using DRL, with an information disclosure scheme that guarantees the privacy of each domain's sensitive details. However, the proposed formulation doesn't optimize latency, which is a critical requirement for many of the 5G use cases. Furthermore, VNE is different from SFC placement as the latter considers the order between the VNFs, which can affect the end-to-end latency. In [16], The authors employ DRL for VNFFG embedding on multiple non-cooperative domains, where the domains compete for the VNFFG requests and update their pricing policies accordingly, while the client trains its own model using the observed QoS metrics of the deployed SFCs using an actor-critic policy gradient algorithm. However, the model is only tested with a small network setup of 3 domains and with a limited VNFFG length.

In this contribution, we propose a DRL-based model along with a cross-domain framework for the placement of multi-domain SFCs while jointly optimizing cost and latency.

III. Proposed Architecture

In this section, we depict our proposed architectural framework with its components and detail the learning mechanism of the agent and its different interactions. Figure 1 illustrates our proposed architecture, which is composed of the orchestrators of each respective domain, the centralized Multi-Domain Orchestrator (MDO), and the Multi-Domain Interface (MDI). In a multi-domain setting, SFC placement is performed in two phases: First, by the MDO based on the abstracted view of the network, where each VNF is assigned to a domain, and the inter-domain links are selected. Then, based on that placement, the SFC is partitioned into a set of sub-chains that are placed by each selected domain orchestrator with a full view of the underlying network, and linked to the rest of the SFC through the inter-domain network to ensure the end-to-end service. In the following, we describe the main components of our proposed architecture and their interactions.

A. Multi-Domain Orchestrator

In our architecture, the MDO uses a DRL agent to perform its placement. The agent's goal is to determine the policy that maximizes the rewards perceived from the environment (i.e., the optimal policy π_*) by creating associations between actions and states through multiple rounds of trial and error, using the environment's feedback. On the MDO level, we train our agent using Deep Deterministic Policy Gradient (DDPG) [17], which is an off-policy actor-critic method, where the actor-network computes deterministic actions according to the current policy, and the critic network computes the Q-values. The model is trained using separate target policy actor and critic networks that remain fixed during the actor and critic network for more stable learning. As shown in Figure 1, the agent performs 3 separate processes: The decision steps in blue $(1-6)$, the policy networks update steps in red $(7-17)$, and the target policy network update steps in green $(18-19)$.

The agent takes as input the observation and the rewards from the environment. That input is sent to the actor-network so that an action is determined according to the current policy in step 3; then, an exploration noise is generated and added to obtain the action that will be taken by the agent in step 6. The noise is added to enable the exploration. Afterward, the reward and next state are collected from the MDI and sent to the replay buffer and critic network in steps 8 and 9. In parallel, each action, state, next state, and reward is saved into a replay buffer until a certain limit called batch size is reached. Once that limit has been attained, the policy update process steps are triggered: a sample from the replay buffer is randomly chosen to select independent combinations, which avoids correlation effects. This sample is used to update the actor and critic networks of the agent (steps 14 to 17) by computing the Mean Squared Error (MSE) between the critic values and the target critic values of the predicted next states. Finally, once the actor and critic network have been updated, a similar soft update is applied to the target actor and critic networks in steps 18 and 19.

B. Multi-Domain Interface

The MDI collects the information that each domain is disposed to disclose and constructs an abstracted view on the global topology, which will be used to create the state fed to the agent. Typically, the domains disclose the amount of resources that are made available, the unit price, and the inter-domain vertices [18]. In this contribution, we assume that the SFCs are placed VNF by VNF. For each VNF placement action, the MDI returns a partial reward and the updated state; then, once the agent has determined the placement of the whole SFC, the Interface module performs the SFC partitioning accordingly and sends the sub-chains to the selected domains. Afterward, it collects the deployment cost and latency of the locally placed sub-SFCs from the domains. It

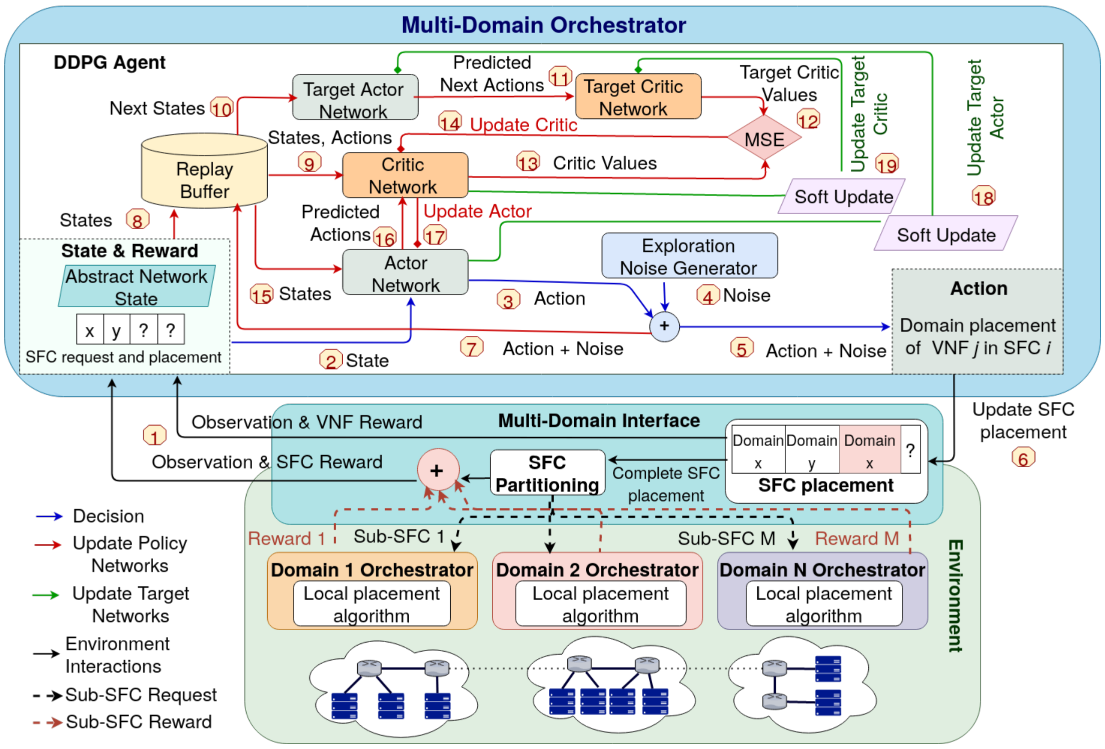

Fig. 1: Multi-domain SFC Embedding Framework

uses them to compute the total reward for the SFC placement and update the environment's state accordingly.

C. Local Domain Orchestrators

Upon receiving their allocated sub-SFCs, the domain orchestrators perform a local placement on a full view of their topology, using their own algorithms, then return the real cost and latency values to the Multi-Domain Interface. Note that the local orchestrator's placement efficiency has an impact on the placement that is performed by the MDO. Indeed, the domains with the least performing algorithms will result in sub-SFCs that are more costly and/or generate more latency, which would lead the agent to adjust its policy to avoid these domains and increase its rewards.

IV. System Model

In this section, we describe the environment's states, actions and rewards, and we formulate the constraints and objectives of the problem as an ILP.

A. States

As explained earlier, due to the lack of information on the Infrastructure Provider's topologies, the MDO disposes of an incomplete view on the state of the network, which means that the environment is Partially Observable for the MDO. The state for each time-step is expressed using information about the VNF currently being placed. Each VNF i from the set of VNFs $\mathcal{V}_i$ of an SFC i is characterized by its set of authorized domains $\mathcal{M}_{i,j}$ on which the VNF can be placed, and its resource requirements $\nabla_{r,i,j}$ for each resource type r from the set $\mathcal{R}$. The states also include the placement of the previous VNFs of the SFC, and specific characteristics of the SFC such as the required amount of bandwidth $\mathcal{W}_i$, the maximal latency ϕ_i^+ and $\mathcal{C}_{i,paid}$ the amount that has been paid by the client to deploy that SFC. Finally, we provide information on the state of the multi-domain topology. The state is composed of the amounts of available resources on each domain n for each resource r and their average unit price denoted by $\mathcal{R}_{r,n}$, and $\zeta_{r,n}$ respectively, and the amounts of available bandwidth, the unit price, and latency for each inter-domain link l from the set $\mathcal{L}$, denoted by $\mathcal{R}_{\omega,l}$, ζ_l and ϕ_l respectively. However, if a domain isn't included in the set of authorized domains for a VNF, its resource values are set to 0 during the VNF's placement time-step.

B. Actions

To perform SFC embedding, the placement of each VNF should be determined. For the sake of simplicity, we assume in our proposal that the shortest paths between the domains for each topology have been pre-computed, which means that the agent only selects the domains where the VNFs have been placed. Therefore, for a VNF j, the number of actions would correspond to the number of its allowed domains $|\mathcal{M}_{i,j}|$, and the associated decision variable of our model is the boolean $\mathcal{X}_{i,j}^n$ which takes the value 1 if the VNF j of SFC i has been mapped to the domain n, and 0 otherwise. To be successfully placed, each SFC i should satisfy the following constraints:

1) Mapping Constraints: Each VNF i can only be placed on one domain from its set of allowed domains, further, two successive VNFs can be mapped to two domains n and m only if there is an available physical path between those domains, which is expressed using the boolean $\rho_{n,m}$:

$$\sum_{n\in\mathcal{M}_{i,j}} \mathcal{X}^n_{i,j} = 1, \qquad \forall j \in \mathcal{V}_i \tag{1}$$

$\forall j \in \mathcal{V}_i, \forall n \in \mathcal{M}_{i,j}, \forall m \in \mathcal{M}_{i,j+1}$:

$$\mathcal{X}^n_{i,j} \cdot \mathcal{X}^m_{i,j+1} \leq \rho_{n,m}, \tag{2}$$

2) Capacity Constraints: To place a VNF j on a certain domain n, the latter must dispose of sufficient amounts of all of the resource types r (CPU, RAM, disk space...) to host it. Therefore, for an SFC placement to succeed, the following constraint must be satisfied:

$$\sum_{j\in\mathcal{V}_i} \nabla_{r,i,j} \cdot \mathcal{X}^n_{i,j} \leq \mathcal{R}_{r,n}, \qquad \forall n \in \mathcal{M}_{i,j+1}, \forall r \in \Re \tag{3}$$

Similarly, for each pair of VNFs j and $j+1$, each link l that is part of the path between their selected domains must dispose of sufficient remaining bandwidth capacity. Denoting by the boolean $\tau^{n,m}_l$ whether a link is part of the physical path between domains n and m, the constraint can be expressed as follows: $\forall l \in \mathcal{L}$:

$$\sum_{j\in\mathcal{V}_i} \sum_{n\in\mathcal{M}_j} \sum_{m\in\mathcal{M}_{j+1}} \mathcal{W}_i \cdot \mathcal{X}^n_{i,j} \cdot \mathcal{X}^m_{i,j+1} \cdot \tau^{n,m}_l \leq \mathcal{R}_{\omega,l} \tag{4}$$

3) Latency Constraint: The end-to-end latency for an SFC i cannot exceed the limit for its selected SLA, to meet the QoS expectations of the client.

$$\phi_i \leq \phi^+_i \tag{5}$$

To compute the end-to-end latency ϕ_i, we must first compute the total inter-domain link latency ϕ^W_i:

$$\phi^W_i = \sum_{j\in\mathcal{V}_i} \sum_{l\in\mathcal{L}} \sum_{n\in\mathcal{M}_{i,j}} \sum_{m\in\mathcal{M}_{i,j+1}} \phi_l \cdot \mathcal{X}^n_{i,j} \cdot \mathcal{X}^m_{i,j+1} \cdot \tau^{n,m}_l \tag{6}$$

And also add ϕ^L_i the sum of latencies ϕ_i, s that would result from the local placement of each sub-SFC s from the set $\mathcal{S}_i$ of sub-chains for an SFC i on their selected domains.

$$\phi^L_i = \sum_{s\in\mathcal{S}_i} \phi_i, s \tag{7}$$

Note that the values ϕ_i, s are collected by the MDI from each local domain orchestrator once each sub-SFC has been placed.

4) Cost Constraint: The total deployment cost, which is composed of the computational and link cost denoted by $\mathcal{C}_{comp}$ and $\mathcal{C}_{link}$ should remain below a certain amount to guarantee a profit percentage of at least γ for the SFC provider:

$$\mathcal{C}_{link} + \mathcal{C}_{comp} \leq (1-\gamma) \cdot \mathcal{C}_{i,paid} \tag{8}$$

Where $\mathcal{C}_{comp}$ and $\mathcal{C}_{link}$ are computed as follows:

$$\mathcal{C}_{comp} = \sum_{n\in\mathcal{N}_d} \sum_{r\in\Re} \sum_{j\in\mathcal{V}_i} \zeta_{r,n} \cdot \nabla_{r,i,j} \cdot \mathcal{X}^n_{i,j} \tag{9}$$

$$\mathcal{C}_{link} = \sum_{j\in\mathcal{V}_i} \sum_{l\in\mathcal{L}} \sum_{n\in\mathcal{M}_j} \sum_{m\in\mathcal{M}_{j+1}} \zeta_l \cdot \mathcal{W}_i \cdot \mathcal{X}^n_{i,j} \cdot \mathcal{X}^m_{i,j+1} \cdot \tau^{n,m}_l + \sum_{s\in\mathcal{S}_i} \zeta_i, s \tag{10}$$

Note that when computing the total link cost, we include the sum of link costs ζ_i, s from deploying the sub-SFCs on the local domains, which is also collected by the MDI.

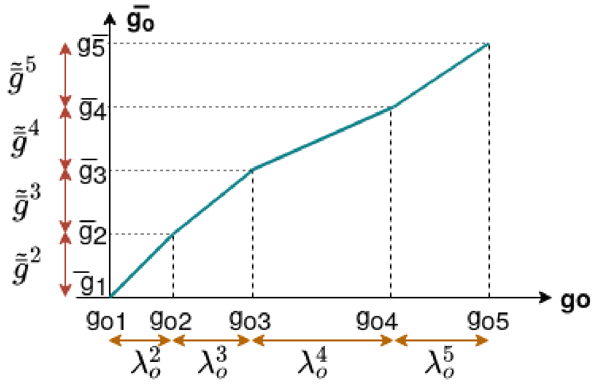

Fig. 2: Linear Physical Programming Class Function

C. Rewards

The goal of this model is to perform multi-domain SFC placement while minimizing deployment cost $\mathcal{C}_i$, the end-to-end latency ϕ_i, and the request rejection rate. Therefore, the rewards for placing an SFC depend on how close the cost and end-to-end latency values are to the optimal ones for the SFC's SLA. To evaluate the quality of the solution, we use Linear Physical Programming [3], which is a Multi-Objective Optimization method that evaluates each objective using preference ranges and outputs a normalized value that reflects the solution's compliance with the preferences of the Decision Maker. For each objective o, we specify k classes of preference (e.g., Highly Undesirable, Undesirable, Acceptable, Desirable, Highly Desirable) where the ranges for each class depend on the SLA of the SFC. Using the range boundary values, for objectives that need to be minimized, the piece-wise linear class functions $\bar{g}_o$ are formulated as follows:

$$\bar{g}_o = \begin{cases} \bar{g}_{o,k-1} + \tilde{\bar{g}}^k \left(\frac{g_o - g_{o,k-1}}{\lambda^k_o}\right) & \text{If } g_o \in [g_{o,k-1}, g_{o,k}], 2 \leq k \leq 5 \\ 0 & \text{If } g_o < g_{o,1} \end{cases} \tag{11}$$

As illustrated in Figure 2, $g_{o,k}$ represent the upper x-axis limit for the preference range k and objective o, and $\bar{g}_k$ and $\tilde{\bar{g}}^k$ represent the y-axis upper limit and amplitude for the preference range k, note that these last values are the same across objectives, which is equivalent to normalization. The interval limits on the y-axis are computed as follows:

$$\bar{g}_1 = 0, \quad \bar{g}_2 = \tilde{\bar{g}}^2 \text{ is a small positive number (e.g 0.1)} \tag{12}$$

$$\tilde{\bar{g}}^k = \beta(n_{sc}-1)\tilde{\bar{g}}^{k-1}; (3 \leq k \leq 5); n_{sc} > 1; \beta > 1 \tag{13}$$

Where β is a convexity parameter that can be determined by solving the following inequality system [19] :

$$\begin{cases} \beta > 1 \\ \beta > \frac{g_{i,k}}{g_{i,k-1}(n_{sc}-1)}; \quad (3 \leq k \leq 5) \end{cases}$$

The reward function for an SFC is a fixed placement reward minus a penalty which is inversely proportional to the quality of the solution, expressed as the sum of the class functions for each objective. Indeed, if the objectives reach the optimal value, their class function value would be 0, but when cost and latency increase, their class function and thus their related penalty also increases. Therefore, the final reward function that the agent aims to maximize can be formulated as follows:

$$reward = \mathcal{A}_i \cdot (R_S - \bar{g}_{\mathcal{C}} - \bar{g}_{\phi}) - (1 - \mathcal{A}_i) \cdot P_F \qquad (14)$$

With R_S being the reward obtained for successfully placing the SFC, P_F the penalty for failing to place that SFC, and $\mathcal{A}_i$ the boolean that expresses whether the SFC has been placed.

V. Evaluation

In the following, we depict the evaluation testbed and settings for our proposed solution. Our solution is implemented on a physical machine with 4 Core i7-5500U 2.40GHz CPU Cores and 8GB of memory, hosting an Ubuntu 18.04 x64 Operating System. The simulation environment is implemented using Python. To learn the optimal policy π_*, we use PyTorch to implement fully-connected Deep Neural Networks (DNN) for the actor, critic, and target actor and critic with hidden layers of 400 and 300 nodes applying the Hyperbolic Tangent (*tanh*) activation function in the output layer and the Rectified Linear Unit (ReLU) activation function in the two hidden layers. We have also used layer normalization between the hidden layers to enable smoother gradients, faster training, and better generalization accuracy. We apply a discount factor γ of 0.99, and the learning rates of the actor and critic networks are set to 10^{-5} and 10^{-3}, respectively. Furthermore, the target network update coefficient τ is set at 0.001 with a batch size of 64. The neural network parameters are updated using the ADAM optimizer [20], and the Ornstein-Uhlenbeck process is employed to generate the exploration noise. As for the local placement algorithms by the domains, we implement a Genetic algorithm in Python that refines its solutions using solution generation, mutation, and crossover.

A. Model Training

To train our model, we run 5000 independent episodes. For each episode, we randomly generate multi-domain topologies of 8 domains using the *networkx* library, where each local domain topology has a 3-level Fat-Tree structure with 16 servers. We also generate SFC requests of 4-10 VNFs, where each VNF is characterized by a type (small, medium, large) that defines the amounts of required resources for each resource type. Each request specifies the number of users, the allowed domain sets, and the SLA that the SFC must satisfy. Table I illustrates the cost and latency preference values for each SLA, as defined in [3], based on the 5G service types, where the SLA classes 0 and 1 correspond to the low latency use cases with different bandwidth requirements, the SLA classes of 2-5 represent the classic internet usage scenario with different service levels, while the last SLA 5 corresponds to the best effort service level. We denote by - the *Unacceptable* SLA class, by H-U the *Highly Undesirable* SLA class, by U the *Undesirable* class, by T the *Tolerable* class and by D and H-D the *Desirable* and *Highly Desirable* classes respectively. For each episode, 10 successive SFCs must be placed with

SLA Class	**-**	**H-U**	**U**	**T**	**D**	**H-D**
Latency (ms)						
0	Strict latency requirement : $<$ 150, high bandwidth required					
1	Strict latency requirement : $<$ 150, low bandwidth required					
2	$>$ 175	165-175	155-165	145-155	135-145	$<$ 135
3	$>$ 210	190-210	180-190 ms	170-180	170-150	$<$ 150
4	$>$ 315	285-315	270-285	255-270	225-255	$<$ 225
5	No latency requirement					
Relative Deployment Cost (%)						
All SLAs	>95	85-95	70-85	60-70	50-60	<50

TABLE I: Preference Ranges by SLA for each objective

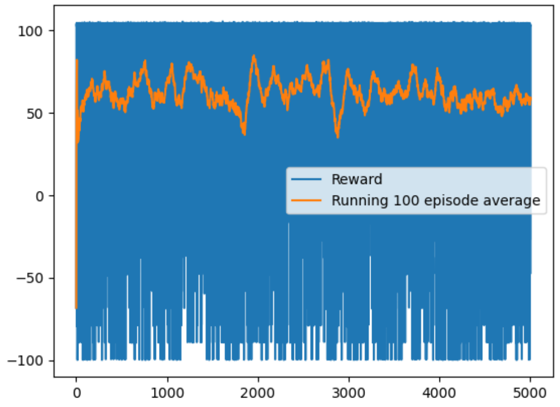

Fig. 3: DDPG Model Training Reward

a maximum reward of 10 for each successfully placed SFC minus SLA-related penalties, summed with the partial rewards of placing each VNF. If one SFC placement fails, the episode is terminated, the resources of the topology are freed, and the agent receives a penalty of -100. Figure 3 illustrates the rewards and running average of 100 episodes for our solution during the learning phase of the agent. It can be observed that the rewards quickly converge to higher values, with occasional placement failures that can be imputed to the exploration actions that are performed by the agent. Further, early SFC placement failures have a higher impact on the reward. Indeed, the first failure in an episode leads to its termination, thus preventing the following SFCs of the episode from being placed and reducing the total reward.

B. Model Evaluation

Once our model has been trained, it is evaluated using two key metrics. First, we observe the acceptance rate of the requests to assess the model's ability to capture the problem's constraints. The second evaluation metric is related to the quality of the obtained solution, where the cost and latency of each accepted request are compared to the optimal values of its SLA. For this part of the experiment, we generate 5000 new episodes and switch the agent to the full exploitation mode. Figure 4 shows the results of the model's evaluation. The first sub-figure in blue represents the running average of the individual SFC request rejection rate. It can be seen that the rejection rates oscillate between 0 and 3% and stabilize

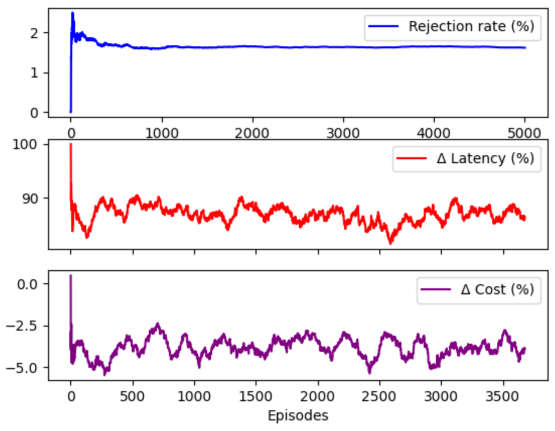

Fig. 4: DDPG Model Evaluation Results

under 2%, note that due to the fact that the rejection of one SFC in the episode leads to its termination, it increases the rejection rate per episode by 74% of successful episodes while in total, more than 97% of the processed SFC requests have been placed successfully. Afterward, we observe the SFCs that have been successfully placed and assess their latency and cost according to the selected SLA. The second and third sub-figures in red and purple display the 100 episodes running average difference between the obtained latency and cost for each episode, and their optimal values, respectively. Since our objective is to minimize cost and latency, a positive difference would mean that the obtained results are below (thus better than) the optimal. The obtained results show that the latency values are on average around $80 - 90\%$ below the optimal value for that SFC, and the cost values are on average $2-5\%$ above the optimal value of 50% of the SFC's revenue but remain largely below the maximum 95% value. Therefore, it can be concluded that our proposal's SFC placements achieve latency values that reach the optimal value for their SLAs while maximizing the profit margin of the SFC providers.

VI. Conclusion

In this paper, we devised a model based on Reinforcement Learning to optimize the placement of multi-domain SFCs, while satisfying a set of service-related requirements. We proposed a multi-layered architecture, and modeled our system to capture the environment's states and transitions. Then, we trained a DDPG agent to perform the multi-domain placement, using Physical Programming to generate rewards that reflect the quality of the chosen actions. Our evaluation results demonstrated the efficiency of our proposal with SFC request rejection rates of under 2%, and cost and latency values that are close to optimal. For future works, we plan to extend this proposal to support post-deployment SFC orchestration.

Acknowledgment

This work has been partially supported by the European Union's H2020 MonB5G (grant no. 871780) project.

References

[1] T. Taleb, A. Ksentini, M. Chen, and R. Jäntti, "Coping with emerging mobile social media applications through dynamic service function chaining," *IEEE Trans. Wirel. Commun.*, vol. 15, no. 4, pp. 2859–2871, 2016.

[2] B. Nour, A. Ksentini, N. Herbaut, P. A. Frangoudis, and H. Moungla, "A blockchain-based network slice broker for 5g services," *IEEE Networking Letters*, vol. 1, no. 3, 2019.

[3] N. Toumi, O. Bernier, D.-E. Meddour, and A. Ksentini, "On using physical programming for multi-domain sfc placement with limited visibility," *IEEE Transactions on Cloud Computing*, pp. 1–1, 2020.

[4] R. Boutaba et al., "A comprehensive survey on machine learning for networking: Evolution, applications and research opportunities," *Journal of Internet Services and Applications*, vol. 9, 05 2018.

[5] N. C. Luong, D. T. Hoang, S. Gong, D. Niyato, P. Wang, Y. Liang, and D. I. Kim, "Applications of deep reinforcement learning in communications and networking: A survey," *IEEE Communications Surveys Tutorials*, vol. 21, no. 4, pp. 3133–3174, 2019.

[6] Q. Zhang, X. Wang, I. Kim, P. Palacharla, and T. Ikeuchi, "Service function chaining in multi-domain networks," in *2016 Opt. Fiber Commun. Conf. (OFC)*, March 2016, pp. 1–3.

[7] G. Sun, Y. Li, D. Liao, and V. Chang, "Service function chain orchestration across multiple domains: A full mesh aggregation approach," *IEEE Transactions on Network and Service Management*, vol. 15, no. 3, pp. 1175–1191, Sep. 2018.

[8] S. Yang, F. Li, S. Trajanovski, R. Yahyapour, and X. Fu, "Recent advances of resource allocation in network function virtualization," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 2, pp. 295–314, 2021.

[9] N. Jalodia, S. Henna, and A. Davy, "Deep reinforcement learning for topology-aware vnf resource prediction in nfv environments," in *2019 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, 2019, pp. 1–5.

[10] Y. Xiao, Q. Zhang, F. Liu, J. Wang, M. Zhao, Z. Zhang, and J. Zhang, "Nfvdeep: Adaptive online service function chain deployment with deep reinforcement learning," in *2019 IEEE/ACM 27th International Symposium on Quality of Service (IWQoS)*, 2019, pp. 1–10.

[11] H. A. Shah and L. Zhao, "Multi-agent deep reinforcement learning based virtual resource allocation through network function virtualization in internet of things," *IEEE Internet of Things Journal*, pp. 1–1, 2020.

[12] S. Troia, R. Alvizu, and G. Maier, "Reinforcement learning for service function chain reconfiguration in nfv-sdn metro-core optical networks," *IEEE Access*, vol. 7, pp. 167 944–167 957, 2019.

[13] G. Kibalya, J. Serrat, J. Gorricho, R. Pasquini, H. Yao, and P. Zhang, "A reinforcement learning based approach for 5g network slicing across multiple domains," in *2019 15th International Conference on Network and Service Management (CNSM)*, 2019, pp. 1–5.

[14] A. I. Swapna et al., "Policy controlled multi-domain cloud-network slice orchestration strategy based on reinforcement learning," in *2020 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, 2020, pp. 167–173.

[15] D. Andreoletti, T. Velichkova, G. Verticale, M. Tornatore, and S. Giordano, "A privacy-preserving reinforcement learning algorithm for multi-domain virtual network embedding," *IEEE Transactions on Network and Service Management*, vol. 17, no. 4, pp. 2291–2304, 2020.

[16] P. T. A. Quang, A. Bradai, K. D. Singh, and Y. Hadjadj-Aoul, "Multi-domain non-cooperative vnf-fg embedding: A deep reinforcement learning approach," in *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2019, pp. 886–891.

[17] T. P. Lillicrap et al., "Continuous control with deep reinforcement learning," in *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2016.

[18] M. Shen, K. Xu, K. Yang, and H. H. Chen, "Towards efficient virtual network embedding across multiple network domains," in *2014 IEEE 22nd Int. Symp. of Quality of Serv. (IWQoS)*, May 2014, pp. 61–70.

[19] X. Ma and B. Dong, "Linear physical programming-based approach for web service selection," in *2008 International Conference on Information Management, Innovation Management and Industrial Engineering*, vol. 2, 2008, pp. 398–401.

[20] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015.