Multi-domain non-cooperative VNF-FG embedding: A deep reinforcement learning approach

Quang Tran Anh Pham, Abbas Bradai, Kamal Deep Singh, Yassine Hadjadj-Aoul

Multi-domain non-cooperative VNF-FG embedding: A deep reinforcement learning approach

Pham Tran Anh Quang*, Abbas Bradai†, Kamal Deep Singh‡, Yassine Hadjadj-Aoul*
*Inria, Univ Rennes, CNRS, IRISA, France
Email: quang.pham-tran-anh@inria.fr, yassine.hadjadj-aoul@irisa.fr
†XLIM, University of Poitiers, Poitiers, France
Email: abbas.bradai@univ-poitiers.fr
‡Laboratoire Hubert Curien, Université de Saint-Etienne, France
Email: kamal.singh@univ-st-etienne.fr

***Abstract*—Network Function Virtualization (NFV) and service orchestration simplify the deployment and management of network and telecommunication services. The deployment of these services require, typically, the allocation of Virtual Network Function - Forwarding Graph (VNF-FG), which implies not only the fulfillment of the service's requirements in terms of Quality of Service (QoS), but also considering the constraints of the underlying infrastructure. This topic has been well-studied in existing literature, however, its complexity and uncertainty unveil many challenges for researchers and engineers. This issue is especially complex when it comes to placing a service on several non-cooperative domains, where the network operators hide their infrastructure to other competing domains. In this paper, we address these problems by proposing a deep reinforcement learning based VNF-FG embedding approach. The results provide insights into behaviors of non-cooperative domains. They also show the efficiency of proposed VNF-FG deployment approach having automatic inter-domain load balancing.**

I. Introduction

Network Function Virtualization (NFV) promises to reduce the cost of deployment while allowing the operation of large network infrastructures. It provides the possibility to migrate complex network functions from dedicated hardware appliances to general purpose computing, storage, and networking solutions. This transition is also expected to provide significant benefits to the 5G mobile network architecture, by allowing flexible and scalable provisioning of new applications and network services. Moreover, since NFV allows multiple network services to share the same physical infrastructure, it enables new business models and/or economies of scale. In particular, the Network–as–a–Service (NaaS) business model is expected to play a pivotal role in 5G mobile networks, allowing Mobile Network Operators (MNOs) to tap into new revenue streams. In fact, virtualization will allow MNOs to abstract their physical network infrastructure into service specific slices, possibly operated by different mobile virtual network operators (MVNOs) [1] or over the top (OTT) providers.

Virtualization and *adaptive* network service orchestration are two of the main technical enablers that will allow Infrastructure Providers (InPs) to cope with the diverse range of requirements. This will characterize future applications and services. It is worth noticing that in this case, an InP could either be a traditional MNO that decides to open-up its networks to third parties or a new actor in the value chain that focuses only on the deployment and operation of the network infrastructure (the InP could even be a consortia of MNOs). Although a rich body of literature exists on VNF placement [2], virtual network embedding [3], and component placement [4], most of these works focus on the problem of mapping an input virtual network request onto a physical virtualized network substrate.

Machine learning (ML) techniques have been adopted, showing breakthroughs, in a number of application areas. One of the advantages of ML is that it can deal with complicated problems; thus it is intuitive to exploit ML in the network domain where the complex problems are common [5]. ML algorithms can be classified into three categories: supervised learning (SL), unsupervised learning (USL), and reinforcement learning (RL). While SL and USL focus on classification or regression tasks, RL algorithms can help in decision making and can learn to identify the best action policy in order to maximize a given objective function.

Early on, RL techniques have been exploited to solve routing optimization [6]. They are also used in QoS routing [7]; however, table-based RL agents cannot provide efficient solutions for unseen network states and simply do not work in realistic environments with large actions space, which could even be infinite. Deep Reinforcement Learning (DRL) is able to cope with these issues, while overcoming the iterative improvement process of optimization and heuristics by having a DRL agent providing a near-optimal solution in one single step [8]. Moreover, the performance of DRL has been improved by recent breakthroughs [9], [10]; thus paving a way for adopting DRL in the context of networking.

In this paper, we study VNF-FG deployment over non-cooperative multiple domains. Each domain employs a DRL agent to learn from history of its actions and rewards. It learns a local action policy in terms of proposed prices for using its resource. A given client who requests VNF-FG embedding receives these prices and decides the final mapping between the VNFs and the substrate nodes plus links of the domains.

The rest of the paper is structured as follows. Section II reviews VNF-FG embedding and DRL in networking. Section III provides an overview of VNF-FG embedding problem. A DRL-based non-cooperative approach to deploy VNF-FGs in multi-domains is proposed in Section IV. The performance of the proposed approach is verified by simulations, presented in Section V. Finally, Section VI concludes our work.

II. Related works

The need to dynamically deploy services on demand, through VNF-FG embedding, is identified as the core technology of 5G and post-5G networks. Therefore, this issue has been at the very centre of academic and industrial research in recent years. In this section, we review VNF-FG embedding works in single and multi-domains, as well as the use of DRL in networking and especially in VNF-FG embedding.

A. VNF-FG embedding

Different existing solutions for single domain VNF-FG embedding can be classified into two: one stage or two/multi-stages VNF-FG embedding. In [11], [12], authors model the problem as an ILP and propose heuristics to map the virtual graph components on the physical resources, on the basis of a single stage. The main goal of these works is to maximize the revenue generated for the infrastructure providers in terms of bandwidth, computing, storage, radio and/or energy consumption. In [11], authors propose a heuristic based on a backtracking mechanism, while authors in [12] take advantage of the feedback of mapping the VNF-FG to optimize the VNF-FG design. The objective of this work is to optimize the total bandwidth consumption.

The second class of single domain VNF-FG embedding solves the problem in two stages. Such that, the virtual nodes are first mapped based on certain criteria (mainly on the available resources), then the links are mapped in a second stage. Authors in [13] propose to rank and map the nodes based on a Markov random walk algorithm. They take into consideration the availability as well as the neighbors of the underlying nodes in their solution. Authors in [14] propose to dynamically map the virtual links into physical links based on the network conditions of the underlying network.

Recently, a few works tackled the multi-domain VNF-FG embedding [15], [16]. In [16], the authors adopted distributed orchestrator based solutions where the local view of each orchestrator is replicated in the other orchestrators to build a global view of the network. Problems of complexity and confidentiality can be raised in such architectures. [15] studied the cooperation between domains to determine the global optimal VNF-FG embedding. The original problem is converted into a factor graph and each domain control a part of this graph which relates to its networks. By adopting this approach, the convergence speed and the quality of solution can be enhanced.

B. Deep-reinforcement learning in networking

With Reinforcement Learning (RL), an agent learns by interacting with its environment. The agent learns to perform the best action for each state by performing actions and observing the rewards or penalties. Given enough observations, an optimal policy can be learned. Thus, the training data in reinforcement learning is a set of state-action-rewards. Some algorithms for RL are: Q-learning, State Action Reward State Action (SARSA), Deep Q-Network [17], etc.

Recently, some works proposed RL-based approaches for VNF-FG embedding [18][19]. In [18], authors map the VNF-FG in two stages: node mapping stage then link mapping stage. They propose two approaches for link mapping: MaVEnM and MaVEn-S, based on Multi-Commodity Flow algorithm and the shortest-path algorithm, respectively. Regarding the node mapping, the authors propose an MDP based approach. The node mapping algorithm is called for each node embedding request. Consequently, this approach is time consuming and not adapted for real time embedding. Authors in [19] propose a multi-agent based reinforcement learning approach for virtual network embedding. These agents evaluate the feedback to learn the best policy to adopt in order to optimally allocate the required resources to the virtual nodes and links. However, they only tackle single domain scenario.

To the best of our knowledge, we are the first to propose a RL-based approach for multi-domain VNF-FG embedding.

III. VNF-FG embedding problem

In this paper, we consider a system with a number of VNF-FGs. Each VNF-FG consists of VNFs connected by virtual links. Each VNF requests a specific amount of resources, which mainly depends on the number of supported users and the type of services. The resources requested by VNFs could be the amount of central processing units (CPUs), Random Access Memory (RAM), Storage, Radio, etc. The number of resource types is K_{VNF}. Each type of resource is indexed by $0, 1, \ldots, K_{\text{VNF}}-1$. We denote $h_{n',k}$ as the amount of resource k requested by VNF n' and $\mathbf{h}_{n'} = [h_{n',0}, ..., h_{n',K_{\text{VNF}}-1}]$ as the vector of requested resources of VNF n'. Without loss of generality, we consider the core network where computing tasks are major; thus the requests of VNFs are computing resources (i.e. CPU, RAM, and storage) ($K_{\text{VNF}} = 3$). Note that it is expected that the proposed mechanism can be exploited with more complicated VNF resources, by extending the input of the DRL as discussed in next section. The VNFs are connected by virtual links (VLs). Each VL is characterized by K_{VL} network oriented metrics (e.g. bandwidth, latency, loss rate, etc.). The request of metric k of VL l' is denoted as $h_{l',k}$ and $\mathbf{h}_{l'} = [h_{l',0}, \ldots, h_{l',K_{\text{VL}}-1}]$ as the vector of request resources of VL l'. We consider bandwidth, latency, and packet loss rate as the metrics of VLs ($K_{\text{VL}} = 3$).

Let us denote the set of VNFs and VLs as $\mathcal{N}'$ and $\mathcal{L}'$, then the number of VNFs and the number of VLs are $|\mathcal{N}'|$ and $|\mathcal{L}'|$, respectively. The sets of nodes and links of the substrate network are $\mathcal{N}$ and $\mathcal{L}$, therefore the number of substrate nodes and substrate links are $|\mathcal{N}|$ and $|\mathcal{L}|$, respectively. A VNF is successfully deployed when its host has enough resources, i.e.

$$\sum_{n'} \phi_n^{n'} h_{n',k} \leq r_{n,k}, \forall n, k \tag{1}$$

where $\phi_n^{n'}$ is a binary variable which is 1 if n' is deployed at substrate node n and $r_{n,k}$ is the available amount of resource k at substrate node n. Note that each VNF is deployed at only one substrate node n; thus

$$\sum_{n} \phi_n^{n'} \leq 1, \forall n'. \tag{2}$$

A VL is successfully deployed when its ends (VNFs) are deployed and its QoS requirements are met. We have

$$\sum_{l'} \phi_l^{l'} h_{l',bw} \leq r_{l,bw}, \forall l, \tag{3}$$

$$h_{l',delay} \leq D(\phi^{l'}) \tag{4}$$

$$h_{l',loss} \leq R(\phi^{l'}) \tag{5}$$

where $\phi_l^{l'}$ is a binary variable which is 1 if l' is deployed at the substrate link l, $r_{l,bw}$ is the available amount of bandwidth at the substrate link l, and $D(\phi^{l'})$ and $R(\phi^{l'})$ are the actual latency and loss rate corresponding to a given mapping $\phi^{l'} = \left[\phi_0^{l'}, \phi_1^{l'}, ..., \phi_{|\mathcal{L}|-1}^{l'}\right]$.

The requests of VNF-FGs are described by a 3D-array of $|\mathcal{N}'| \times |\mathcal{N}'| \times (2 \times K_{\text{VNF}} + K_{\text{VL}})$. This 3D-array is a $(2 \times K_{\text{VNF}} + K_{\text{VL}})$-channel of $|\mathcal{N}'| \times |\mathcal{N}'|$ matrices. The first K_{VL} channel describes VLs requirements. The next $2 \times K_{\text{VNF}}$ channels express the resource requests of the source and destination VNFs respectively. As convolutional layers are able to extract mutual impacts between links in a network (shown in [20]), this form of VNF-FG requests enables us to apply convolutional layers to learn the mutual impacts between VLs.

VNF-FG requests are sent by the client who is the owner of the VNF-FG as the `state` to different domains. Then, each domain inputs it into a DRL network in order to determine an action which expresses the mapping of VNF-FGs to its substrate network. Even though we can define a discrete action space for this problem, its size could be enormous. For instance, each VL can be mapped to an ordered subset of substrate links, thus we have the number of actions for only mapping VLs is $|\mathcal{L}'| \times \sum_{k=0}^{k=|\mathcal{L}|} \frac{|\mathcal{L}|!}{(|\mathcal{L}|-k)!}$. In [20], the continuous link weights were adopted as the action space of routing problems since it reduces the size of action space to $|\mathcal{L}|$ and improves the performance of learning process. Then, the deep deterministic policy gradient (DDPG) method was utilized to deal with continuous action space thanks to its performance [21]. We exploit this idea to our problem and the action space becomes the prices proposed to the client to satisfy the request.

The price of resource m at substrate node n of VNF n' is denoted as $c_{n,n'}^m$. The price vector of VNF n' is $\mathbf{c}_{n'} = \left[\mathbf{c}_{0,n'}, \mathbf{c}_{1,n'}, ..., \mathbf{c}_{|\mathcal{N}|-1,n'}\right]$, where $\mathbf{c}_{i,n'} = \left[c_{n,n'}^0, ..., c_{n,n'}^{K_{\text{VNF}}-1}\right]$. Therefore, it needs $|\mathcal{N}'| \times |\mathcal{N}| \times K_{\text{VNF}}$ entries in the action to express the prices of computing resources. Moreover, we need $|\mathcal{L}'| \times |\mathcal{L}|$ entries in the action to express the price of using a unit of bandwidth of substrate links for each VL. The price vector of VL l' is $\mathbf{c}_{l'} = \left[\mathbf{c}_{0,l'}, \mathbf{c}_{1,l'}, ..., \mathbf{c}_{|\mathcal{L}|-1,l'}\right]$, where $c_{i,l'}$ is the price of a unit of bandwidth for VL l'. These prices are determined by each domain, which are then sent to the client. We limit the lowest price in order to guarantee it is greater or equal to the actual deployment cost. The client who wants to deploy VNF-FGs observes the prices proposed by domains and makes a final decision so as to minimize its deployment cost. This process will be detailed in next section.

IV. Multi-domain non-cooperative VNF-FG embedding

A. The framework

In this section, we propose a framework for multi-domain non-cooperative VNF-FG embedding. As a non-cooperative framework, each domain does not have information of the topology as well as the available resources of other domains. Moreover, they do not communicate to each other to determine the global optimal solution. Each domain only exposes the prices of resources to the client, then the client makes a final decision based on these prices. Fig. 1 describes the framework that we propose in this paper.

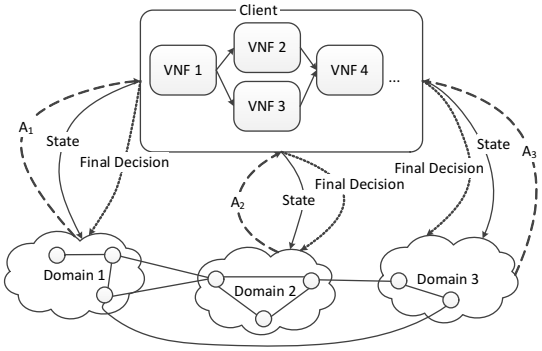


Fig. 1: Non-cooperative Multidomain VNF-FG embedding framework

Recall, the state is a 3D-array representing VNF-FG requests, which is input to the DRL module present in the domain. The action determined by DRL is the price sent by a domain to the client. The client then decides where to finally embed the VNF-FG as the final action. The final action also determines the reward to different domains bidding to embed the VNF-FG. The framework comprises rounds of 3 steps:

- Step 1: The client who owns the VNF-FGs translates the requests of VNF-FGs (CPU, RAM, Storage, bandwidth, latency, and loss rate) to a 3D-array and sends it to domains. It represents the `state` of VNF-FGs. We assume that a unique request of VNF-FGs arrives and is also finally embedded in each round if resources are available.
- Step 2: Each domain uses `state` to compute the action through the actor network. Note that the action of domain i ($\mathbf{A}_i$) is the price proposed to the client which in turn is a function of the cost of using its resources, i.e., $c_{n,n'}^k, \forall n \in \mathcal{N}_i$ and $c_{l,l'}, \forall l \in \mathcal{L}_i$, where $\mathcal{N}_i$ and $\mathcal{L}_i$ represent, respectively, the sets of nodes and links of the substrate network of the domain i.
- Step 3: A `Final Decision` is made by the client and domains are informed about it. Each domain follows the `Final Decision` to deploy VNFs and VLs and this final action also results in a reward for the domain.

B. Deep RL agent model

The proposed approach adopts an off-policy, actor-critic, deterministic policy gradient algorithm (DDPG) [22] RL agent that interacts with the data network through states, actions, and rewards [17]. The state is the requests of VNF-FGs, which is sent by the client in step 1. Each domain determines the prices of its resources corresponding to the request and sends them to the client (step 2). Thus actions are like bids of domains for selling their resources. Client then makes the `Final Decision`. After receiving the `Final Decision` from

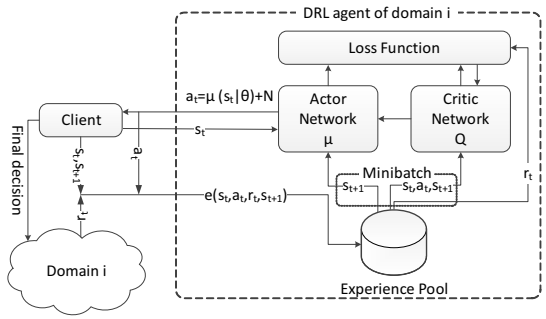
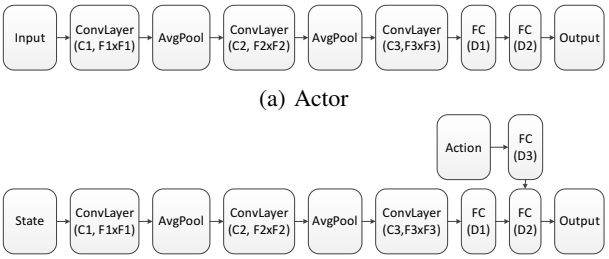


Fig. 2: DDPG in non-cooperative multidomain VNF-FG embedding



(a) Actor

(b) Critic

Fig. 3: Neural networks

the client, domains deploy the VNFs and VLs, accordingly. The client observes the performance and gives the reward to domains according to their offered prices. The reward of domain i can be determined as follows

$$r_i = \sum_{n' \in \mathcal{N}'} \sum_{n \in \mathcal{N}_i} \sum_{k \in K_c} \omega_n^{n'} c_{n,n'}^k h_{n',k} + \sum_{l \in \mathcal{L}_i} \sum_{l' \in \mathcal{L}'} \omega_l^{l'} c_{l,l'} h_{l',bw} \tag{6}$$

where $\omega_n^{n'} = 1$ ($\omega_l^{l'} = 1$) if and only if $\phi_n^{n'} = 1$ ($\phi_l^{l'} = 1$) and the requested resources (QoS) are satisfied.

In DDPG, two neural networks are maintained to learn the policy (the actor network) and the Q-value (the critic network) separately. Fig. 2 presents the operation of DDPG algorithm. The state s_t is observed by the DRL agent of each domain. An action is determined by the actor with its current parameters. This action could be added with noise N to explore the environment better. The critic network assess the advantages of an action and leads the actor to a better policy. The client collects actions of domains (bidding prices), decides a final action and executes it. Each domain receives its rewards in return to providing resources to the client. These rewards are fed into DRL agent to compute the loss and update the parameters of DRL agent.

The objective of domain i is to identify the optimal policy μ_i mapping the requests of VNF-FGs to the local action $\mathbf{A}_i$, $\mu_i : S \rightarrow \mathbf{A}_i$, that maximizes its reward presented in Eq. (6). It is done by repeatedly enhancing its knowledge of the connections between the requests, actions, and rewards through the means of deep neural network consisting in the actor and the critic networks. Meanwhile, the objective of the client is to maximize the number of allocated VNFs and VLs with the lowest cost. A lightweight heuristic algorithm will be discussed in the next section to handle this task. The actor and critic networks are composed of different layers used to extract useful information from the state as shown in Fig. 3. We use three convolutional layers (ConvLayers). The number of filters and the kernel size of ConvLayer i are C_i and (F_i, F_i). In addition, we add average pool layers to reduce the amount of parameters and control overfitting. The output of ConvLayers expresses the mutual impacts between VLs taking into account all properties of VLs. Also, we use Fully Connected (FC) layers to map mutual impacts to the action. The number of units of FC i is D_i. Note that we also convert the action by a FC layer in the critic.

C. Decision maker models

After collecting the prices from the different domains ($\mathbf{A}_i$), the client has to make a decision in order to deploy its VNF-FGs. While it is difficult to guarantee that a path will meet the required QoS of virtual links, it is possible to guarantee the needs of resources of VNFs. Consequently, the decision has to consider the capabilities of substrate nodes when deploying VNFs as well as the cost of deployment. We revise First Fit algorithm, a well-known resource allocation algorithm, into the Cost-based First Fit (`CFF`) algorithm in order to address the needs of a decision.

Alg. 1 describes the `CFF` algorithm. The global action $\mathbf{A}_*$ formed by actions of domains is split into the costs related to VNFs $\mathbf{A}_{*,vnf}$ and the cost related to VLs $\mathbf{A}_{*,vl}$. For VNF i, a dot product between the cost of VNF i ($\mathbf{A}_{*,vnf}[i] = \mathbf{c}_i$) and its request resource ($\mathbf{h}_i$) provides the deployment cost corresponding to each substrate node (line 6). From line 8 to line 12, the substrate node that has sufficient resources and the lowest deployment cost will be selected. The client sends request to the domain $D(j)$ which has the substrate node j. Domain $D(j)$ checks if it can provide the resources and replies to the client. If $D(j)$ accepts the request, `CFF` algorithm moves to the next VNF. Otherwise, it checks the next lowest cost substrate node. `CFF` algorithm continues until all VNFs are considered. The `CFF` algorithm outputs a mapping between VNFs and the substrate nodes. The VNF mapping obtained by `CFF` algorithm identifies the locations of VNFs. To map VL l', we exploit $\mathbf{A}_{*,vl}[l']$ as the weight of links, then executing Dijkstra algorithm to find the lowest cost path of l'.

V. Simulation results

To assess the performance of the proposed approach we use a network topology BtEurope [23] of 24 nodes and 37 full-duplex links. The capacity of links are assigned one of following values randomly: 20 Mbps, 30 Mbps, 50 Mbps or 100 Mbps. The OMNeT++ discrete event simulator [24] (v5.4.1) was used to obtain the latency and packet loss rate to assess the quality of deployed VLs.

For each configuration, we execute 10 simulations with random numbers, sizes, and shapes of VNF-FGs. Each VNF-FG has from 3 to 6 VNFs and the connectivity of 0.5. The requested resources of VNFs are normalized and distributed uniformly. The amounts of available resources of each substrate node are random in the range $(0.3, 2.0)$. Each virtual links arbitrarily requests a bandwidth in range of 1 Mbps to 30

Algorithm 1: Cost-based First Fit Algorithm (CFF)

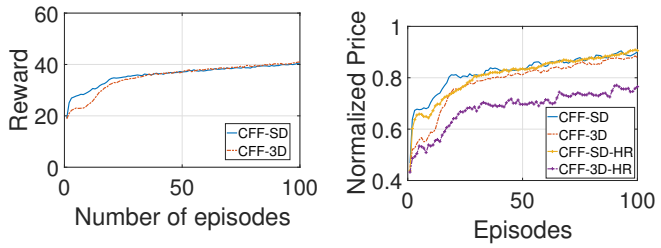
Input: $\mathbf{A}_i, i = 1, ..., D$ from D domains
Output: VNF mapping $\phi_n^{n'}$
Initialize $\phi_n^{n'} = 0, \forall n, n'$ $\mathbf{A}_* \leftarrow [\mathbf{A}_1, ..., \mathbf{A}_D]$,
$\mathbf{A}_{*,vnf}, \mathbf{A}_{*,vl} = \texttt{split}\,[\mathbf{A}_*]$;
foreach *VNF* i **do**
  Compute $\mathbf{P} = \mathbf{A}_{*,vnf}[i] \cdot \mathbf{h}_i$;
  Sort $\mathbf{P}$ in ascending order;
  foreach *entry* j *in* $\mathbf{P}$ **do**
    Send request to $D(j)$;
    if $D(j)$ *accepts request* **then**
      $\phi_j^i = 1$;
      $D(j)$ deploys j at node i;

Mbps, latency of 1 ms to 100 ms, and loss rate of 0% to 0.5%. An Ornstein-Uhlenbeck process is used for the exploration noise [21].

Adam [25] is used for learning the neural network parameters. The learning rates of actor and critic networks are 10^{-4} and 10^{-3}, respectively. The discount factor γ is 0.99. The parameter of target network is updated with the coefficient of $\tau = 0.001$. The batch size is 32. The number of units of fully-connected layers is 300. The numbers of features of 3 ConvLayers are 16, 64, and 128 respectively. The kernel sizes of ConvLayers are $(4, 4)$, $(2, 2)$, and $(5, 5)$.

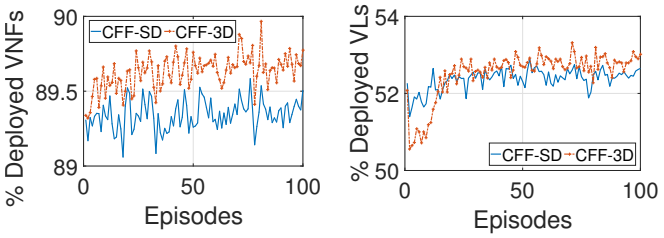
We compare four configurations: single domain with CFF (CFF-SD), three domains with CFF (CFF-3D), simulated annealing algorithm (SA) [26], simulated annealing algorithm with CFF (SA-CFF). First, we compare the average reward of CFF-SD and CFF-3D. The results are presented in Fig. 4a. Generally, both CFF-SD and CFF-3D are able to obtain better reward over time. In CFF-SD, there is no competition; thus the DRL agent can increase the price to the upper limit in order to achieve better rewards. In CFF-3D, each domain attempts to achieve better rewards by competing with other domains to provide resources and increasing the prices. When there are limitations of resources in several domains, there may be only one domain can host VNFs and VLs. Then, the price will increase due to the lack of competitions. We confirm the above claim by considering the average normalized price in two cases: (i) default available resources (amounts of resources of each substrate node in range $(0.3, 2.0)$) and (ii) high available resource (CFF-SD-HR and CFF-3D-HR) where the lower bound of available resources is 1 instead of 0.3. It means each substrate node in high available resource scenario can host at least one VNF. Fig. 4b describes the normalized prices in these scenarios. The price in CFF-SD increases faster than CFF-3D due to the lack of competitions. The normalized price in CFF-3D also increases because of the limitations of resources at substrate nodes. When we loose these limitations, the normalized price is much lower thanks to the competitions between domains (CFF-3D-HR).

Generally, CFF-3D offers lower prices than CFF-SD while providing a slightly better percentages of deployed VNFs and



(a) Rewards of CFF-SDand CFF-3D (b) Average of prices

Fig. 4: Rewards and Normalized Prices



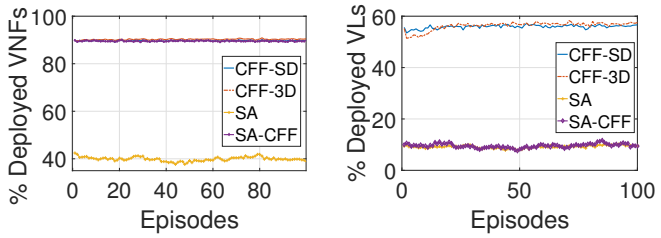
(a) Percentage of Deployed VNFs (b) Percentage of Deployed VLs

Fig. 5: Percentage of deployed VNFs and VLs

VLs as shown in Fig. 5. This may be because each DRL agent in CFF-3D manages a smaller networks compare to the DRL agent in CFF-SD. Consequently, both the state and the action space of DRL agents in CFF-3D are smaller of DRL agents in CFF-SD. The smaller state and action spaces help DRL agents obtain better solutions.

The comparison of the performance of CFF-SD and CFF-3D, with SA and SA-CFF is presented in Fig. 6. Note that we consider single domain scenario when applying simulated annealing. The very large action state of VNF-FG embedding problem causes challenges for simulated annealing method in searching good solutions. Consequently, the percentage of deployed VNFs of SA is much lower than of CFF-3D and CFF-SD. CFF algorithm can help SA in finding the capable substrate nodes; however, it cannot guarantee these VNF placements can lead to a good VL deployment. Consequently, the percentage of deployed VLs of SA and SA-CFF are similar and remarkable lower than of CFF-3D and CFF-SD.

The next simulations are to confirm the load balancing capabilities of DRL-agents. Table I shows the average resources of domains. Domain 1 has the lowest CPU and RAM resources, while domain 3 has the lowest storage resource. Note that

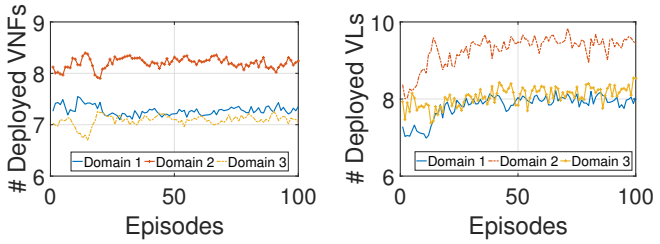


(a) Percentage of Deployed VNFs (b) Percentage of Deployed VLs

Fig. 6: Comparison of CFF-SD, CFF-3D, SA, and SA-CFF

Domain	1	2	3
CPU	7.93	8.14	8.35
RAM	8.21	8.29	8.72
Storage	8.53	7.82	6.95
Capacity	540 Mbps	710 Mbps	570 Mbps

TABLE I: Resources of domains



(a) VNFs of each domain (b) VLs of each domain

Fig. 7: Mean of deployed VNFs and deployed VLs of each domain

the limitation in one of resources may cause the failure in deploying VNFs. Consequently, domain 2 can host more VNFs than other domains as shown in Fig. 7a. The limitations in storage resource of domain 3 prevents it from deploying VNFs, thus it can host less VNFs than domain 1. Fig. 7b shows the mean of deployed VLs over time of each domain. Domain 2 has the best network capacity, thus it can host more VLs than other domains. Domain 3 has slightly better capacity than domain 1; thus its mean of deployed of VLs is a bit better than of domain 1. Generally, the domain with higher resources will host more VNFs and VLs. In other words, DRL-agents are able to balancing the load between them even though they acts selfishly while determining the actions.

VI. Conclusions

In this paper, we studied the multi-domain non-cooperative VNF-FG embedding problem. We introduced a framework in which each domain determines the bidding price of using its own resources selfishly, then the final decision is made by the owner of VNF-FGs by executing `CFF`, a light-weight heuristic algorithm. The results confirmed embedding VNF-FG in non-cooperative multi-domain offers gains to the owner of VNF-FGs by reducing the bidding prices thanks to competitions between domains while the successful deployment of VNFs and VLs is slightly improved. The analysis of each domain confirmed the inter-domain load balancing capabilities in order to achieve the best performance. A thorough study will be conducted in future to compare the performance of this proposed approach with more approaches.

References

[1] A. Nakao and P. Du, "Application-Specific Slicing for MVNO and Traffic Characterization," *J. Opt. Commun. Netw.*, vol. 9, no. 2, pp. A256–A262, Feb 2017.

[2] H. Moens and F. De Turck, "VNF-P: A model for efficient placement of virtualized network functions," in *IEEE CNSM*, Rio, Brasil, 2014.

[3] Z. Despotovic, A. Hecker, A. N. Malik, R. Guerzoni, I. Vaishnavi, R. Trivisonno, and S. A. Beker, "VNetMapper: A fast and scalable approach to virtual networks embedding," in *Proc. of IEEE ICCCN*, Shanghai, China, 2014.

[4] B. Jennings and R. Stadler, "Resource management in clouds: Survey and research challenges," *J. Netw. Syst. Manage.*, vol. 23, no. 3, pp. 567–619, Jul. 2015.

[5] M. Wang, Y. Cui, X. Wang, S. Xiao, and J. Jiang, "Machine learning for networking: Workflow, advances and opportunities," *IEEE Network*, vol. 32, no. 2, pp. 92–99, March 2018.

[6] J. A. Boyan and M. L. Littman, "Packet routing in dynamically changing networks: A reinforcement learning approach," in *Advances in neural information processing systems*, 1994, pp. 671–678.

[7] S. Lin, I. F. Akyildiz, P. Wang, and M. Luo, "Qos-aware adaptive routing in multi-layer hierarchical software defined networks: A reinforcement learning approach," in *2016 IEEE SCC*, June 2016, pp. 25–33.

[8] G. Stampa, M. Arias, D. Sanchez-Charles, V. Muntés-Mulero, and A. Cabellos, "A deep-reinforcement learning approach for software-defined networking routing optimization," *CoRR*, vol. abs/1709.07080, 2017.

[9] Y. Li, "Deep reinforcement learning: An overview," *arXiv preprint arXiv:1701.07274*, 2017.

[10] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "A brief survey of deep reinforcement learning," *CoRR*, vol. abs/1708.05866, 2017.

[11] S. Sahhaf, W. Tavernier, M. Rost, S. Schmid, D. Colle, M. Pickavet, and P. Demeester, "Network service chaining with optimized network function embedding supporting service decompositions," *Computer Networks*, vol. 93, pp. 492 – 505, 2015, cloud Networking and Communications II.

[12] Z. Ye, X. Cao, J. Wang, H. Yu, and C. Qiao, "Joint topology design and mapping of service function chains for efficient, scalable, and reliable network functions virtualization," *IEEE Network*, vol. 30, no. 3, pp. 81–87, May 2016.

[13] X. Cheng, S. Su, Z. Zhang, H. Wang, F. Yang, Y. Luo, and J. Wang, "Virtual network embedding through topology-aware node ranking," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 2, pp. 38–47, 2011.

[14] M. Yu, Y. Yi, J. Rexford, and M. Chiang, "Rethinking virtual network embedding: substrate support for path splitting and migration," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 17–29, 2008.

[15] P. T. A. Quang, A. Bradai, K. D. Singh, G. Picard, and R. Riggio, "Single and multi-domain adaptive allocation algorithms for vnf forwarding graph embedding," *IEEE Transactions on Network and Service Management*, pp. 1–1, 2018.

[16] Q. Zhang, X. Wang, I. Kim, P. Palacharla, and T. Ikeuchi, "Service function chaining in multi-domain networks," in *IEEE OFC*, 2016, pp. 1–3.

[17] R. S. Sutton, A. G. Barto *et al.*, *Reinforcement learning: An introduction*. MIT press, 1998.

[18] S. Haeri and L. Trajković, "Virtual network embedding via monte carlo tree search," *IEEE transactions on cybernetics*, vol. 48, no. 2, pp. 510–521, 2018.

[19] R. Mijumbi, J.-L. Gorricho, J. Serrat, M. Claeys, F. De Turck, and S. Latré, "Design and evaluation of learning algorithms for dynamic resource management in virtual networks," in *IEEE NOMS*, 2014, pp. 1–9.

[20] Q. T. A. Pham, Y. Hadjadj-Aoul, and A. Outtagarts, "Deep Reinforcement Learning based QoS-aware Routing in Knowledge-defined networking," in *EAI Qshine*, Ho Chi Minh City, Vietnam, Dec. 2018, pp. 1–13.

[21] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *CoRR*, vol. abs/1509.02971, 2015.

[22] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *ICML*, 2014, pp. 387–395.

[23] The internet topology zoo. [Online]. Available: http://www.topology-zoo.org/dataset.html

[24] A. Varga, "Discrete event simulation system," in *Proc. of the European Simulation Multiconference*, 2011.

[25] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014. [Online]. Available: http://arxiv.org/abs/1412.6980

[26] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.