Robust Deep Reinforcement Learning Algorithm for VNF-FG Embedding

Abdelmounaim Bouroudi, Abdelkader Outtagarts, Yassine Hadjadj-Aoul

Robust Deep Reinforcement Learning Algorithm for VNF-FG Embedding

Abdelmounaim Bouroudi
Nokia Bell Labs
Nokia Paris-Saclay, France
abdelmounaim.bouroudi@nokia.com

Abdelkader Outtagarts
Nokia Bell Labs
Nokia Paris-Saclay, France
abdelkader.outtagarts@nokia-bell-labs.com

Yassine Hadjadj-Aoul
INRIA, Univ Rennes, CNRS, IRISA
Rennes, France
yassine.hadjadj-aoul@irisa.fr

***Abstract*—Network slicing, also known as the virtual network embedding (VNE) problem, is an NP-hard optimization problem. Compared to traditional approaches, the methods relying on deep reinforcement learning yield better performance without exhibiting issues such as stacking at local minima and/or solutions' space exploration limits. These algorithms present, however, different performances according to the employed approach, and the problem to be treated, resulting in robustness problems. To overcome these limits, we propose the adoption of the best algorithm, from a selection of learning strategies, in terms of reward and sample efficiency at each time step. The proposed strategy acts as a meta-algorithm that brings more robustness to the network by dynamically selecting the best solution for a specific scenario. Our solution proved its efficiency and managed to dynamically select the best algorithm in terms of the best acceptance ratio of the deployed services and outperform all the stand-alone algorithms.**

***Index Terms*—Virtual Network Embedding, Deep Reinforcement Learning, Algorithm Selection, Stationary bandit, B5G/6G**

I. Introduction

The sixth generation of mobile networks is expected to provide a multi-purpose network with massive machine-to-machine communications and a multitude of services with diverse requirements, thereby requiring a new architecture to better meet such needs [1]. To meet such requirements, network providers are adopting new technologies including network function virtualization (NFV) and software-defined network (SDN). These technologies allow the virtualization of the physical infrastructure and the dynamic creation and sharing of network instances and services which enable the concept of network slicing.

The coexistence of different network functions on the same substrate network has led to the NFV placement problem (also known as Virtual Network Embedding "VNE"), in which we look to efficiently allocate resources for network functions an links connecting them (forming a Forwarding Graph). The VNE is an NP-hard problem with several existing solutions [2]. Mathematical optimization is only able to deal with very small network instances [3]. Heuristics converge extremely fast, but reach typically only local minima [4]. Meta-heuristics, which are only applicable to medium-sized networks, suffer from a lack of knowledge accumulation [5].

Given these limitations, machine learning approaches have been explored, and more particularly Deep Reinforcement Learning (DRL), which are more appropriate to deal with the unpredictable behavior of the virtualized services. The authors, in [6], [7], proposed a new DRL strategy based on the DDPG algorithm [8], with improvements in the exploration capacity combined with a heuristic in order to guarantee in the worst case the performance of the latter. The proposed solution provides significantly better results compared to a vanilla DDPG. However, the main drawback of this solution is their lack of robustness, which is a critical aspect to consider in next generation cellular networks [9].

Since existing DRL-based algorithms are case-specific solutions, their performances can be affected by considering new unseen scenarios. This creates a risk to the network making such standalone solutions unsafe. Moreover, these agents present different learning potentials, where some algorithms allow quick learning but prematurely converge to a local optimum, while others learn slowly in the beginning but yield better performance at the end.

To solve these problems, we propose a new robust resource allocation algorithm based on the Epochal Stochastic Bandit Algorithm Selection (ESBAS) framework [10], that we adapt to the context of the VNE problem. ESBAS is an off-line Algorithm Selection (AS) framework for off-policy RL algorithms. Its principle is to select among a set of RL algorithms the one maximizing the expected return. The algorithm splits the execution time to epochs of exponential size, in which the set of algorithms is not updated, and the selection is done using a rebooted stochastic bandit with accumulation of knowledge using a buffer to store the executed episodes. This framework allows both a competitive execution between different RL agents by selecting the most promising agent at each iteration, and a cooperative exploration of the action space through the transition sharing mechanism.

The rest of this paper is organized as follows. We start by a background section, where we present the VNE problem and formulate it as a Markov Decision Process (MDP). Then, we present an offline AS algorithm based on ESBAS. In the fifth section, the simulation results of offline selection are presented alongside with a discussion and a comparison with already existing standalone algorithms. Finally, we provide a conclusion in section V.

II. BACKGOUND

In this section, we introduce the modeling of the VNE problem and its formulation using an MDP, which is common to all the considered learning techniques.

A. VNE Problem Formulation

In the VNE problem we aim to deploy a set of V VNF-Forwarding Graphs (VNF-FG) noted $\mathcal{G} = \{G_1, ..., G_V\}$ into a substrate network. On one hand, Each VNF-FG G_ζ is described by a directed graph with a set of $\mathcal{N}^{'}$ Virtual Network Functions (VNFs) where each VNF $n^{'}$ requests a set of k_{VNF} type of resources represented by a vector $h_{n'} = [h_{n',0}, .., h_{n',K_{\text{VNF}}-1}]$. These VNFs are connected by a set of $\mathcal{L}^{'}$ virtual links and each link $l^{'}$ requests k_{VL} types of resources denoted by a vector $f_{l'} = [l_{n',0}, .., l_{n',K_{\text{VL}}-1}]$. On the other hand, the substrate network is also represented by a directed graph where each node n has a specific amount of each resource k denoted $r_{n,k}$, and each link has also a specific amount of resource k denoted $r_{l,k}$.

To deploy a VNF-FG, we need to deploy all its VNFs and its VLs. A VNF can be deployed on a specific node if this last has sufficient resources for each requested resource and Each VNF is deployed on only one substrate node. The same requirements are applied on the VLs with an aditional condition that to deploy a VLs, all VNFs it connects must be deployed.

We define an auxiliary binary variable g_ζ indicating if the VNF-FG ζ is deployed. The objective is to maximize the number of deployed VNF-FG called the Acceptance Ratio (AR), which is defined by:

$$AR = \frac{\sum_\zeta g_\zeta}{V} \tag{1}$$

B. Markov Decision Process formulation

Reinforcement Learning consist in learning through trial and errors using an agent that interact with an environment [11]. Each time steps t, the agent receives the state of the environment s_t, and based on it, the agent chooses an action a_t to execute on the environment following a policy ($\pi : S \rightarrow A$). The environment return a feedback on the executed action known as reward r_t with the new environment state s_{t+1}. This feedback is used to optimize the policy with the goal of maximizing the expected reward. This process can be modeled as an MDP. In this paper, we use the same MDP formulation presented in our previous paper [7]:

- **State representation**: the environment state is represented by the formulation of the VNF-FGs which are described by a vector of size $|\mathcal{N}^{'}| \times K_{\text{VNF}} + |\mathcal{L}^{'}| \times K_{\text{VL}}$. This vector describes the requests of VNFs and VLs of all VNF-FGs and is fed into the DRL agent as the environment state.
- **Action**: each action a_t is a vector of size $|\mathcal{N}| \times |\mathcal{N}^{'}| + |\mathcal{L}| \times |\mathcal{L}^{'}|$ and $a_t = [a_t^{n,n^{'}}, w_t^{l,l^{'}}]$ where $a_t^{n,n^{'}}, w_t^{l,l^{'}}) \in [0, 1]$ showing the favor of assigning VNF $n^{'}$ to node n (resp. virtual link l to substrate link $l^{'}$) at time-step t. As the action is sparse, we propose using a binary versions of the action vector to convert it to a feasible solution by applying the WF2A heuristic [7].
- **Reward**: to assess the performance of the RL algorithm, the Acceptance Ratio defined in (1) is adopted as a reward measure.

III. A NEW FRAMEWORK FOR DRL-BASED ALGORITHMS' SELECTION

As DRL based algorithms have different performances depending on the instances of the problem, it is relevant to have a meta-algorithm solution allowing the association of each algorithm to an instance of the problem to obtain better performances. This solution is a selection algorithm framework.

Formally, the algorithms' selection is a sequential decision problem $\mathcal{D}$ with a set of instances $\mathcal{I}$, a set of algorithms $\mathcal{A} = A_1, A_2, ..., A_n$ to solve $\mathcal{D}$ and a metric $m : \mathcal{A} \times \mathcal{I} \rightarrow \mathbb{R}$ allowing the qualitative measurement of the performance of each algorithm for a problem instance [12]. The goal is, thus, to define a selector $\mathcal{S}$ that maps each instance $i \in \mathcal{I}$ of the problem to an algorithm $\mathcal{S}(i) \in \mathcal{A}$ that allows the optimization of the overall performance according to the metric m.

In our case, $\mathcal{D}$ is the VNE problem, the algorithms are the different DRL agents, and the metric is the Acceptance Ratio returned by the agent. Constructing such a selection algorithm allows the combination of the strengths of multiple heterogeneous agents (algorithms). This is achieved through a competition and a cooperation mechanisms [13]. The competitive aspect appears in the execution between different RL agents by selecting the most promising agent at each iteration, while the cooperative aspect concerns the exploration of the action space through the transition sharing mechanism. In the following subsections, we first describe the unifying framework for RL agents. Then we introduce the proposed offline algorithm for the selection of the best agent.

A. Unifying view for Reinforcement Learning agents

To apply the Algorithm Selection framework for RL agents, we propose a unified view for the different agents independently of their learning mechanism. This unified view allows the Algorithm Selection solution to deal with these agents as black boxes and select the best one, relying only on their returns. For this purpose, we assume that we have a set of off-policy RL algorithms called *portfolio* that learn from the same shared data-set (fairness assumption). This framework was first introduced in [10] for general purpose RL algorithms. In this paper, we adapt this framework to the VNE problem and we propose the workflow presented in Fig 1. The process starts with a selection of an algorithm A_i from the portfolio (1). In (2), the selected algorithm is used in the execution environment to generate a trajectory ϵ_τ (3) and returns a cumulative reward $\mu(\epsilon_\tau)$ (4.2). On one hand, the generated trajectory (4.1) is stored in a shared memory (buffer) that will be used to update all the algorithms in the portfolio (5.2). On the other hand, the cumulative reward is used to update the Algorithm Selection policy corresponding to the selected algorithm (5.1). The final

global goal is to optimize the expected cumulative reward defined as:

$$\mathop{\mathbb{E}}_{S}\left[\sum_{\tau=1}^{T}\mu(\varepsilon_\tau)\right] \tag{2}$$

Based on these assumptions, and using these formulations, we propose in the next section two meta-algorithms which allow the selection of DRL agents to solve the VNE problem in two manners, offline selection and online selection with attention.

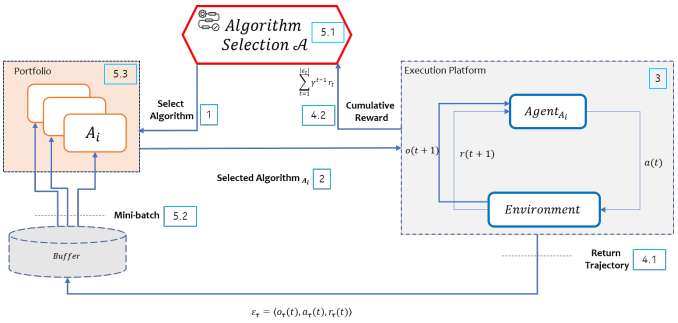

Fig. 1. Algorithm Selection Flow Diagram.

Algorithm 1: ESBAS for VNE problem

Input: δ_0, A, G, T

1 $g \leftarrow 0$
2 **for** $\beta \leftarrow 0, \infty$ **do**
3 **for** $A_k \in A$ **do**
4 **if** $g \geq G$ **then**
5 $\pi^k_{\delta_{2^\beta-1}}$: learn π^k on trajectories of epoch 2^β
6 $n \leftarrow 0, \forall A_k \in A, n_k \leftarrow 0, x_k \leftarrow 0$
7 **for** $\tau \leftarrow 2^\beta, 2^{\beta+1}$ **do**
8 $A_{k_{max}} = \text{argmax}_{A_k}\left(x_k + \sqrt{\xi\frac{log(n)}{n_k}}\right)$
9 $\mu \leftarrow 0$
10 **for** $t \in 0, T$ **do**
11 Execute a^k_t generated by $A_{k_{\max}} \rightarrow r_t, s_{t+1}$
12 $(s_t, a^k_t, r_t, s_{t+1}) \rightarrow \delta$
13 $\mu \leftarrow \mu + r_t$
14 $g \leftarrow g + 1$
15 $x_{k_{max}} \leftarrow \frac{n_{k_{\max}} x_{k_{\max}} + \mu}{n_{k_{max}}}$
16 $n_{k_{\max}} \leftarrow n_{k_{\max}} + 1$
17 $n \leftarrow n + 1$

B. Offline Algorithm Selection

Based on [10], we propose an adapted version of Epochal Stochastic Bandit Algorithm Selection (ESBAS) framework which is a meta-algorithm used to solve the off-policy RL Algorithm Selection problem by considering it as an Exploration/Exploitation problem. In their work, Authors relied on the Upper Confidence Bound (UCB1) algorithm [14] to assure a good trade-off between exploration and exploitation of the different agents' performances and exploration abilities. The main idea behind ESBAS is to block the update for all agents during exponential sized epochs in order to create a stationary system allowing the selection using UCB1.

In Alg. 1 we present the ESBAS algorithm used for VNE problem. In this algorithm, we have two main parts: The first part, from line 2 to 5, the algorithm starts by updating all agent's policies using the past recorded trajectories only if we have enough records in the reply buffer (threshold G). The second part starts from the 6th line in which we initialize the UCB1 parameters. From line 7 to 17 we execute UCB1 on exponential sized epochs. During the epoch, the meta-algorithm computes the optimistic guess using UCB1 formula in line 8 and generates a trajectory using the selected algorithm from line 10 to 14 (in VNE problem, a trajectory is a number T of placements). In each trajectory we calculate the accumulated reward by the executed agent, and we store the new record to the buffer δ. The last step, from line 15 to 17, allows the update of the different parameters of the UCB1 algorithm.

IV. Simulation Results

To simulate the different algorithms, we had first to setup the DRL parameters as mentioned in [7]. For the substrate network, we use the BtEurope network topology [15] with 24 nodes and 37 full-duplex links. For the VNF-FG request we use 10 connected graphs generated using the Erdos-Renyi model [16]. In table I, we present the amount of resources available on substrate network, as well as the resource values requested randomly by the VNF-FGs. The portfolio consists of three off-policy DRL agents, we used three Off-policy DRL agents already used for VNE problem. Each agents have Actor and Critic Networks. Each network is a Dense layer with 300 hidden units each and a ReLU activation function. For the output we use a Sigmoid activation function for the Actor Network and a linear activation function for Critic Networks.The description of the different agents is as follow:

- **W2FA_DDPG agent**: which is the ordinary DDPG algorithm the W2FA heuristic [7].
- **EEDDPG agent**: which is the implement the EEDDPG in [7] without the evolutionary operation.
- **EEDDPG Evolution agent**: which is the full version of EEDDPG with evolutionary operation.

Resource types	Available Resources	Requested Resources
CPU	12-79.8	1, 2, 4, 8
RAM	128-1024	2, 4, 8, 16, 32
Storage	1000-10000	1, 5, 20, 100
Bandwidth	1000-10000	10, 20, 50, 100

TABLE I
Network Resources and VNF-FG Requests

To test the performances of our solution, we used the OMNeT++ simulation tool to deploy the topology based on the same platform used by authors in [7]. ESBAS algorithm

is run for 10 exponential-sized epochs with $\beta = [\![0, 9]\!]$. Fig. 2 shows the Acceptance Ratio of the ESBAS algorithm compared with the three agents. All algorithms are executed on epochs of exponential size, where the updates are run only at the beginning of the epoch. Taking the three agents without ESBAS, the figure shows that the best agent in the large epochs is EEDDPG Evolution, and this is coherent with what authors mentioned in [7], since EEDDPG with Evolution has the possibility to learn early and quickly reaches good Acceptance Ratio. In exponential epochs, updates are less frequent, and the other agents don't perform as well as EEDDPG with Evolution. Compared with ESBAS, we can easily notice that ESBAS outperforms all the tree agents in all the big epochs starting from the third one.

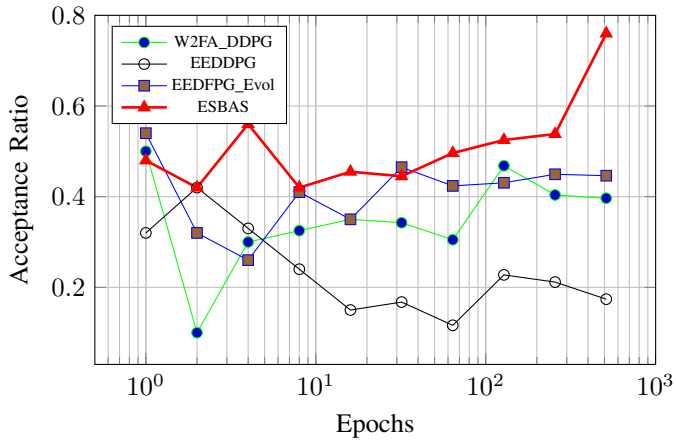

Fig. 2. Acceptance Ratio of ESBAS Compared with DDPG, EEDDPG and EEDDPG Evolution.

In Fig. 3 we present the selection decisions of ESBAS described by the algorithm usage percentage. We can notice that ESBAS makes bad selection decisions in the first small epochs, and afterward starting from the 5th epoch, it starts selecting EEDDPG and EEDDPG with Evolution. With larger epochs ESBAS makes good selection choices, and we notice that both W2FA_DDPG and EEDDPG are vanishing while EEDDPG with evolution replace them more than 80% in the last epoch.

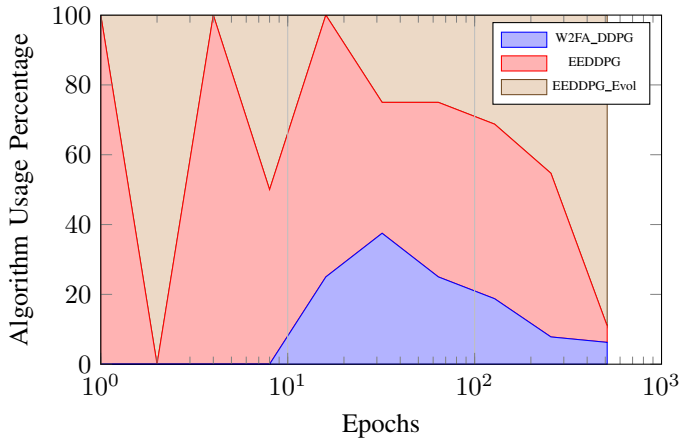

Fig. 3. Selection Ratio of ESBAS.

V. Conclusion

The problem of virtual network embedding is one of the most important challenges in B5G networks. Due to the heterogeneous nature of future networks and the unpredictable behavior of its services, this problem has been addressed using DRL algorithms. In this paper, we adopt the ESBAS framework, which takes advantage of different DRL algorithms, to solve the VNE problem. We propose an adapted version of the offline algorithm selection framework in which the selection is performed in epochs of exponential size during which the agents' updates are blocked. This technique allows the creation of stationary system to ensure a good algorithm selection. We showed the superiority of our approach over standalone DRL algorithms, and highlighted the robustness of the solution and it's ability to select the best algorithm dynamically depending on the considered scenario.

References

[1] A. Dogra, R. K. Jha, and S. Jain, "A survey on beyond 5g network with the advent of 6g: Architecture and emerging technologies," *IEEE Access*, vol. 9, pp. 67 512–67 547, 2020.

[2] E. Amaldi, S. Coniglio, A. M. Koster, and M. Tieves, "On the computational complexity of the virtual network embedding problem," *Electronic Notes in Discrete Mathematics*, vol. 52, pp. 213–220, 2016.

[3] Z. Allybokus, N. Perrot, J. Leguay, L. Maggi, and E. Gourdin, "Virtual function placement for service chaining with partial orders and anti-affinity rules," *Networks*, vol. 71, no. 2, pp. 97–106, 2018.

[4] S. Khebbache, M. Hadji, and D. Zeghlache, "Scalable and cost-efficient algorithms for vnf chaining and placement problem," in *2017 20th conference on innovations in clouds, internet and networks (ICIN)*. IEEE, 2017, pp. 92–99.

[5] ——, "A multi-objective non-dominated sorting genetic algorithm for vnf chains placement," in *2018 15th IEEE Annual Consumer Communications & Networking Conference (CCNC)*. IEEE, 2018, pp. 1–4.

[6] P. T. A. Quang, Y. Hadjadj-Aoul, and A. Outtagarts, "A deep reinforcement learning approach for vnf forwarding graph embedding," *IEEE Transactions on Network and Service Management*, vol. 16, no. 4, pp. 1318–1331, 2019.

[7] ——, "Evolutionary actor-multi-critic model for vnf-fg embedding," in *2020 IEEE 17th Annual Consumer Communications & Networking Conference (CCNC)*. IEEE, 2020, pp. 1–6.

[8] 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," *arXiv preprint arXiv:1509.02971*, 2015.

[9] C. Li, W. Guo, S. C. Sun, S. Al-Rubaye, and A. Tsourdos, "Trustworthy deep learning in 6g-enabled mass autonomy: From concept to quality-of-trust key performance indicators," *IEEE Vehicular Technology Magazine*, vol. 15, no. 4, pp. 112–121, 2020.

[10] R. Laroche and R. Feraud, "Reinforcement learning algorithm selection," *arXiv preprint arXiv:1701.08810*, 2017.

[11] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of artificial intelligence research*, vol. 4, pp. 237–285, 1996.

[12] R. John, "Rice. the algorithm selection problem. volume 15 of advances in computers," 1976.

[13] H. Zheng, J. Jiang, P. Wei, G. Long, and C. Zhang, "Competitive and cooperative heterogeneous deep reinforcement learning," in *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS*, 2020.

[14] M. N. Katehakis and A. F. Veinott Jr, "The multi-armed bandit problem: decomposition and computation," *Mathematics of Operations Research*, vol. 12, no. 2, pp. 262–268, 1987.

[15] S. Knight, "The internet topology zoo," 2010. [Online]. Available: http://www.topology-zoo.org/dataset.html

[16] P. Erdős, A. Rényi *et al.*, "On the evolution of random graphs," *Publ. Math. Inst. Hung. Acad. Sci*, vol. 5, no. 1, pp. 17–60, 1960.