SCHEMA III: Dynamic & scalable VNE framework based on Multi-Agent RL for 5G/6G networks

Anestis Dalgkitsis*, Luis A. Garrido*, Kostas Ramantas*, John S. Vardakas*
George Kormetzas‡ and Christos Verikoukis§
*Iquadrat Informatica S.L., Barcelona, Spain,
§Computer Engineering & Informatics Dept., University of Patras, Greece
‡Information and Communication Systems Engineering Dept., University of the Aegean, Samos, Greece
Email: {a.dalgkitsis, l.garrido, kramantas, jvardakas}@iquadrat.com, gkorm@aegean.gr and cveri@ceid.upatras.gr

***Abstract*—Network Virtualization (NV) has proved a promising technology that allows multiple heterogeneous Virtual Networks (VNs) to operate simultaneously on the same infrastructure. Dynamic Virtual Network Embedding (NVE) has emerged as an enabler of elasticity and scalability in the VN deployment and resource allocation of the physical infrastructure. However, the key challenge in realizing NV in a sustainable way is how to dynamically embed VNs efficiently into physical network, which is defined as the VNE problem. To address this challenge, this paper proposes an approach that leverages Multi-Agent Reinforcement Learning (MARL) to solve the dynamic VNE of VNs for 5G/6G communication systems. The proposed framework consists of multiple horizontally distributed RL agents that co-operate to devise temporally dynamic VNE placements. The key contributions of this work are introducing a novel dynamic VNE orchestration framework for multi-domain networks based on Distributed RL, providing a scalable VNE framework targeted to Ultra-Reliable Low-Latency Communication (URLLC) services, evaluating and comparing the proposed algorithm with existing solutions in the state of the art. The paper concludes that there is a significant improvement in latency of 144.191% when compared to the baselines.**

***Index Terms*—Dynamic Virtual Network Embedding (VNE), Multi-Agent Reinforcement Learning (MARL), Network Virtualization (NV), Zero-touch Orchestration and Management (ZSM).**

I. Introduction

With the increasing demand for faster and more efficient networks, Network Virtualization (NV) has emerged as a promising technology for improving the performance of networks. NV allows for multiple heterogeneous Virtual Networks (VNs) to run on top of the same infrastructure, improving the utilization of physical network resources and avoiding any kind of inefficient utilization of the infrastructure. However, the key challenge in realizing NV is how to efficiently embed Virtual Networks into physical networks, defined as the virtual network embedding (VNE) problem.

The efficient management of network architecture is crucial for the development of Beyond-5G (B5G) and 6G communication systems. In this context, VNE is a promising technology that allows Infrastructure Providers (InPs) and Internet Service Providers (ISPs) to decouple the underlying physical network resources and services. The ISPs propose virtual network requests (VNRs) for network services, while the InPs are responsible for deploying and maintaining the network equipment and allocating physical resources to the VNRs. This decoupling of resources and services enables efficient scalability of the network architecture.

979-8-3503-1090-0/23/$31.00 © 2023 IEEE

Algorithmically, the VNE is an NP-hard problem, which means it can take an exponential amount of time in the worst case. As a result, the best way to approach optimization in an NP-hard problem is through heuristics, meta-heuristics, and approximation algorithms, with the fastest to converge being the best. Fortunately, such system optimization problems can be effectively modeled as Markov Decision Processes (MDPs). Artificial Intelligence (AI) with MDP-based solutions, such as Reinforcement Learning (RL), has dominated various industries in recent years. Not only has AI with MDP-based solutions defeated human players in board and video games, but it has also found numerous applications in the industry, such as self-driving cars and robotics, as demonstrated in recent literature and research [1], [2].

In this paper, we propose, design, develop and evaluate a framework capable to solve the VNE problem in multiple domains, dynamically in real-time with Distributed RL. The proposed framework consists of multiple distributed RL agents that co-operate horizontally to devise temporally dynamic VNE placements. The key contributions of this work can be summarized as follows:

- Introducing a novel dynamic VNE orchestration framework for multi-domain, multi-tenant networks, based on Multi-Agent Reinforcement Learning (MARL).
- Providing a scalable VNE framework targeted to Ultra-Reliable Low-Latency Communication (URLLC) services.
- Evaluating and comparing the proposed algorithm with existing solutions in the state of the art. Our analysis primarily focuses on the performance metrics, with a particular emphasis on URLLC services.

The remainder of this work is organized as follows: Section II presents an in-depth analysis of the relevant literature in the field. Section III provides an outline of the System Model and the architecture. Section IV provides an in-depth analysis of the proposed solution. Section V demonstrates the experimental setup and evaluates the effectiveness of the proposed

approach through simulation. Finally, Section VI concludes this work and discusses our future research directions.

II. Related Work

Effective resource management in B5G and 6G networks is a key area of focus for the recent literature. One of the major challenges in this regard is the placement and mapping of virtual network functions (VNFs) and service function chains (SFCs) through virtual network embedding (VNE). In the context of VNE research, there are distinct components that need to be addressed.

As the AI became more mature and started to outperform humans in several repetitive and simple tasks, the researchers started working with it.

The authors in [3], propose a Continuous-Decision VNE scheme, based on RL (CDRL). They compare the performance of CDRL against two traditional heuristic algorithms and an RL-based embedding algorithm, proving superior performance overall. Similarly, in [4], Yan et al. propose a fusion of graph-based Neural Network structures and Deep RL to generate VNEs With a multi-objective reward function. The idea of mixing Deep Learning (DL) and RL can drastically improve the perceptual performance as stated in [5]. The simulation results show better performance than other SoTA, with up to 39.6% and 70.6% improvement on acceptance ratio and average revenue.

The major shortcoming of the AI and continuing ML, RL is that they are usually operate in a centralized fashion. Although beneficial in terms of cost, centralization can cause irreversible damage and disallows scalability, as it marks a central point of failure or congestion. To counteract this issue, the authors in [6], utilized Horizontal Federated Learning (HFL), a subset of Hierarchical Federated Learning to a multi-domain VNE algorithm. Their proposed solution combines distributed training locally in the domains and parameter sharing among them.

In this work we continue on the idea of MARL to build a totally scalable and distributed VNE algorithm, based on our previous projects and experiences. The proposed framework is a succession of our previous works [7], [8], [9], and [10].

III. System Model & Architecture

The system model in Fig. 1 represents an ISP network distributed over a large geographical area, comprised of multiple cloud domains n and servers u. The network VNEs s consist of multiple chained VNFs.

A. Infrastructure Resources

In an NFV-enabled network, physical network resources are crucial for accommodating its functions. Two nodes are represented by the parameters u and v, connected by the physical link uv. These nodes are of two types, one for network switching and one for NFV-enabled computing. The finite network resources are represented as follows: C_{uv}^{bw} represents the capacity of the network link uv; C_u^{cpu} represents the number of available CPU cores of computing node u; C_u^{ram} represents the amount of available memory of computing node u; and C_u^{hdd} represents the available storage space of node u.

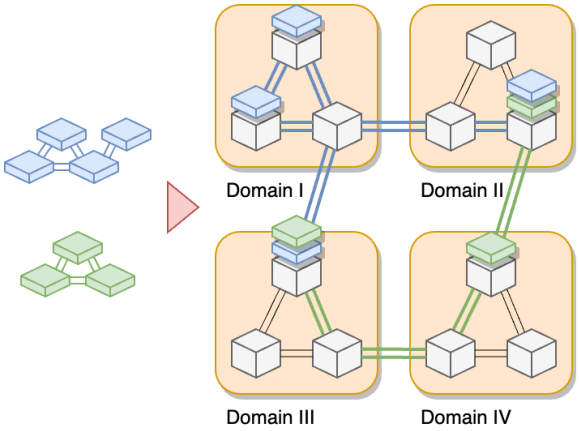

Figure 1. Schematic illustration of the multi-domain VNE, orchestrated by the SCHEMA III framework.

B. Virtual Networks

The Virtual Networks (VNs) s can be represented as unidirectional graphs $G_s = (\mathcal{V}_s, \mathcal{E}_s)$, where the vertices represent VNFs and the edges are the virtual links.

C. Virtual Network Data Flows

The data flows between the VNFs of the VN s are represented as lists $u_{source} \rightarrow u_1 \dots u_j \rightarrow u_{destination}$. The data flows traverse through the intermediate nodes $u_1 \dots u_j$ according to the *Dijkstra* algorithm. The variables u_s, v_s represent two nodes in G_s, while $\mathcal{E}_s$ represents the links $u_s v_s$ that connect neighboring VNFs, served by the nodes u_s and v_s in G_s. The goal of the framework is to embed the VNs in the physical network infrastructure as VNEs.

The data flows of the VNs s, exhibit a distinctive traversal pattern that precludes splitting or reversal and defined as in the following equation 1:

$$\sum_{u \in \mathcal{V}} \sum_{u_s v_s \in \mathcal{E}_s} (z_{uv}^{u_s v_s} - z_{vu}^{u_s v_s}) = \begin{cases} -1, u \text{ is the Source} \\ 0, u \text{ is an Intermediate} \\ 1, u \text{ is the Destination} \end{cases} \tag{1}$$

where $z_{uv}^{u_s v_s}$ and $z_u^{u_s v_s}$ are either 0 or 1 indicating whether $u_s v_s$ traverses the link uv and computing node u respectively.

D. Virtual Network Embedding

The utilization of a physical link uv by the VNE s necessitates the traversal of the interconnected nodes u and v, which can be formulated as follows in 2:

$$z_u^{u_s v_s} z_v^{u_s v_s} = \begin{cases} 1, z_{uv}^{u_s v_s} = 1 \\ 0, \text{otherwise} \end{cases} \tag{2}$$

E. VNE Orchestration

During orchestration the VNEs s are categorized into three discrete classes:

1) Deployed VNEs $\mathcal{S}$: A VNE is considered fully deployed when the VN mapping satisfies all conditions of placement in the infrastructure nodes.

2) Redirected VNEs $\mathcal{R}$: Redirected VNEs occur when the VN mapping violates the service SLA for this mapping, defined as ϕ_s^{delay}, or when there are inadequate resources in the nodes to host the service VNFs.

3) Terminated VNEs $\mathcal{T}$: A VNE is considered terminated when there are no available resource to be deployed or terminated by end of life cycle.

As a general rule of the VNE orchestration, at any time interval the total allocation of bandwidth and computational resource consumption cannot exceed the available infrastructure resources. This constraint applies to all links denoted by $\forall uv$, nodes denoted by $\forall u$, and VNFs denoted by $\forall m$, and is described by the following set of limitations in the equations 3, 4, 5 and 6:

$$\sum_{u_s v_s \in \mathcal{E}_s} \sum_{s \in \mathcal{S} \cup \mathcal{T} \cup \mathcal{R}} \phi_s^{bw} z_{uv}^{u_s v_s} \leq \omega_{uv}^{bw} C_{uv}^{bw}, \tag{3}$$

$$\sum_{u_s v_s \in \mathcal{E}_s} \sum_{s \in \mathcal{S} \cup \mathcal{T} \cup \mathcal{R}} \phi_s^{cpu} p_u^m \leq C_u^{cpu}, \tag{4}$$

$$\sum_{u_s v_s \in \mathcal{E}_s} \sum_{s \in \mathcal{S} \cup \mathcal{T} \cup \mathcal{R}} \phi_s^{ram} z_u^{u_s v_s} \leq \omega_u^{ram} C_u^{ram}, \tag{5}$$

$$\sum_{u_s v_s \in \mathcal{E}_s} \sum_{s \in \mathcal{S} \cup \mathcal{T} \cup \mathcal{R}} \phi_s^{hdd} z_u^{u_s v_s} \leq \omega_u^{hdd} C_u^{hdd}, \tag{6}$$

where the variable p_u^m is 1 if VNF m is placed in node u. The variables $z_{uv}^{u_s v_s}$ and $z_u^{u_s v_s}$ have binary values, as denoted in the previous section, to indicate whether $u_s v_s \in \mathcal{E}_s$ traverses the link uv and node u respectively. The variable ϕ_s^{bw} represents the bandwidth consumption of the physical links due to the mapping of the VNE s on these links. $\phi_{s,m}^{cpu}$, $\phi_{s,m}^{ram}$ and $\phi_{s,m}^{hdd}$ are used for the computational resources that the VNE s VNFs occupy. Finally, ϕ_s^{delay} indicates the maximum tolerated delay by the predefined SLA, the maximum VNE latency in our case.

F. Delay Model & Network Policy Management

We denote latency in links and other elements as follows: d_{uv} the delay of link uv, d_u the delay caused by the computing node hardware and d_m the delay caused by the VNF overlay. Instantiated and running VNEs s in the network show an E2E delay to the users of the VNE $u \in \mathcal{U}_s$, which is highly dependent on the VNF placement and the number of hops. This delay can be expressed as the sum of all delays and cannot exceed its SLA delay ϕ_s^{delay} as indicated:

$$\sum_{uv \in \mathcal{E}} \sum_{u_s v_s \in \mathcal{E}_s} d_{uv} z_{uv}^{u_s v_s} + \sum_{u \in \mathcal{V}} \sum_{u_s v_s \in \mathcal{E}_s} d_u z_u^{u_s v_s} + \sum_{m \in \mathcal{M}} \sum_{u_s \in \mathcal{V}_s} d_m z_m^{u_s} \leq \phi_s^{delay} \tag{7}$$

The inability to achieve the inequality of the equation moves the VNE $s \in \mathcal{S}$ from operating to redirected $s \in \mathcal{R}$ before the end of the current iteration. It triggers a cost penalty used in the reward function to adjust the future actions of the local RL agents.

The E2E service delay D_s during the current interval can be calculated as follows:

$$D_s = \sum_{uv \in \mathcal{E}} \sum_{u_s v_s \in \mathcal{E}_s} d_{uv} z_{uv}^{u_s v_s} + \sum_{u \in \mathcal{V}} \sum_{u_s v_s \in \mathcal{E}_s} d_u z_u^{u_s v_s} + \sum_{m \in \mathcal{M}} \sum_{u_s \in \mathcal{V}_s} d_m z_m^{u_s}, \tag{8}$$

$$\mathbb{D} = \sum_{s \in \mathcal{S} \cup \mathcal{R}} D_s. \tag{9}$$

where D_s is the sum of all mapped links, computing nodes hardware and VNFs overlay delays that are traversed by the chain. The total service delay $\mathbb{D}$ of the current iteration can be expressed as the E2E service delay D_s of all operating VNEs.

IV. Implementation & Designing of the SCHEMA III algorithm

The aim of this section is to provide a comprehensive analysis of the SCHEMA III orchestration framework's hierarchy. Our focus will be on two key components: Firstly, we will describe the intra-domain VNE mapping system, which is responsible for assigning VNF to the internal domain servers. Secondly, we will delve into the inter-domain VNE orchestration system, namely the Voting System, which is based on our prior research works, most importantly [8].

A. VNE Orchestrator Architecture

Regarding the local VNE mapping system, the RL agents are responsible for the creation of a VNE mapping locally in the individual domains. The use of a set of distributed RL agents throughout the infrastructure for the individual domains enables asynchronous operation that avoids unnecessary VNF migrations.

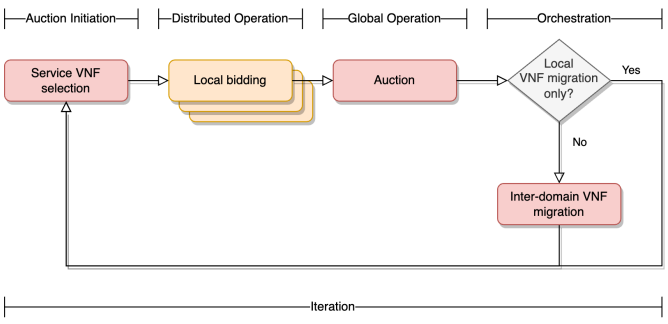

Figure 2. Distributed RL Voting System flow.

Briefly, the components and decision flow, as shown in Fig. IV-A, are the following:

1) Domain: This component represents the environment that the agent is interacting with, modeled as a standard RL problem. It provides interaction with the local domain n part of the network infrastructure which was described in the previous section.

2) RL Agents: The *RL agent* is continuously trained during the interaction with the *Environment* to improve their future recommended actions for VNF placement.

3) Voting System: The Voting System facilitates inter-domain VNF migration through auction-like function.

B. The RL Agents

We model the intra-domain VNF placement and orchestration problem as an MDP which consists of a *State*, an *Action Space* and a *Reward Function*.

The MDP problem space can be defined as follows:

1) State Space: Consists of the intra-domain computational resources of the domain nodes u_n. The local domain state space S_n is defined as a set that contains the VNF hosting computing requirements, the slice SLA and the domain computing and networking utilization ratio for each network element:

$$S_n = (w_{uv}^{bw}, w_u^{cpu}, w_u^{ram}, w_u^{hdd}, \ldots, \phi_s^{delay}, \phi_s^{bw}, \phi_{s,m}^{cpu}, \phi_{s,m}^{ram}, \phi_{s,m}^{hdd})^T, \tag{10}$$

2) Action Space: The *Confidence Vector* A_n, a vector that contains an offer $\in [-1, 1]$ for each domain server u_n to receive the VNF currently to be placed. The maximum of this local action indicates an intra-domain VNF migration:

$$A_n = (i_0, \ldots, i_{u_n})^T, \tag{11}$$

3) Shared Reward: Consists of cost function expressed in (12). The objective of all domains n is to maximize the sum of the function R^{shared} in every iteration and thus, converging in a common state-action that enables cooperation among the domains:

$$R^{shared} = \frac{1}{w_D * \mathbb{D}} + P, \tag{12}$$

where w_D is the weight of the total slice delay $\mathbb{D}$.

In addition, we define a penalty function that is applied when the SLA described in (7) of the slice s is violated:

$$P = -w_P(D_s > \phi_s^{delay}), \tag{13}$$

where w_P expresses the penalty weight.

C. The Voting System

The Voting System is a system that enables inter-domain VNF migration. It acts as an auctioneer between the domains who bid with their actions to receive a slice VNF to their domain.

The Voting System repeats the following steps in every iteration:

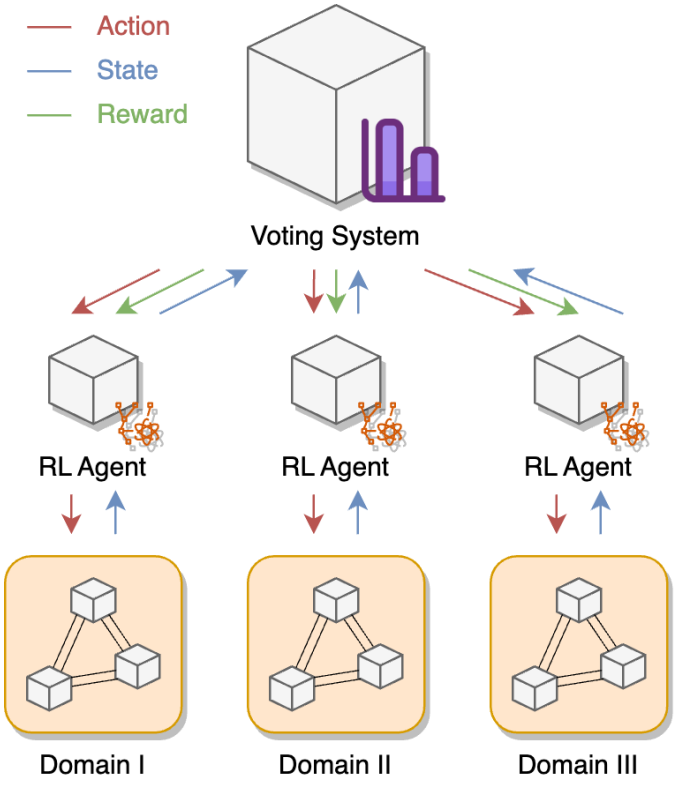

Figure 3. Overview of the SCHEMA III framework architecture in a multi-domain network. All local domains can orchestrate VNFs in parallel and exchange VNFs only during an auction.

1) Selection: The Voting System circulates through the slice VNFs during an *interval*, a predefined time frame. As an auctioneer, it selects the next VNF in round robin fashion for the auction process.

2) Participation: The per domain n Local Agents act as bidders that generate their bid, an action A_n produced by the *Local RL Agent* and secured by the *Safety Shield* as previously described.

3) Auction: The mechanism receives all domain bids and selects the highest bidder (the domain with the highest bid). The candidate domain is notified to initiate the migration.

4) Orchestration: If the candidate domain is not the current domain, the inter-domain migration is initiated. Otherwise, the domain agent performs an intra-domain migration based on the safeRL action to the node with the highest bid.

The Voting System is only responsible for the inter-domain communication. It is non-essential for the local domain orchestration. The Voting System can be deployed quickly as a stateless container at any node of the network, eliminating the single point of failure.

V. Performance Evaluation & Analysis

This following section presents the performance evaluation analysis of the proposed algorithm SCHEMA III via simulation of multiple scenarios and comparison with solutions from the SoTA.

A. Simulation Environment

We have developed an *in situ* VNE simulator previously used in our works [7], [8], [9], and [10]. Similar to our previous implementation, we utilized [11] and the high-level *Keras API* open-source library [12] to build the distributed RL Agent Neural Networks. The RL Agents interface the simulated network via a custom-made OpenAI Gym environment [13] that was developed with Python 3.8.6. For the performance evaluation, we adapted a variation of the *2005 Nordu European network* from *The Internet Topology Zoo* dataset [14] to meet the requirements of our analysis.

B. Comparison Methodology & Baselines

In order to assess the performance of our proposed algorithm, SCHEMA III, we developed a range of VNE algorithms that have been recently proposed in the literature, for the purpose of comparison. Our proposed framework SCHEMA III is compared with:

1) SCHEMA III: is our proposed solution as presented in the previous sections.

2) HFL: is a distributed implementation of VNE with Horizontal Federated Learning (HFL) as presented in various works in the literature, specifically similar to [6].

3) DQN: is a centralized RL-based solution, often used in similar literature in a number of variations. The main point of comparison is scalability and single point-of-failure.

4) Static: is the simplest and most common baseline in the literature. It is also usually employed as a solution in real deployments due to its minimal cost [15], [16].

C. Performance Evaluation

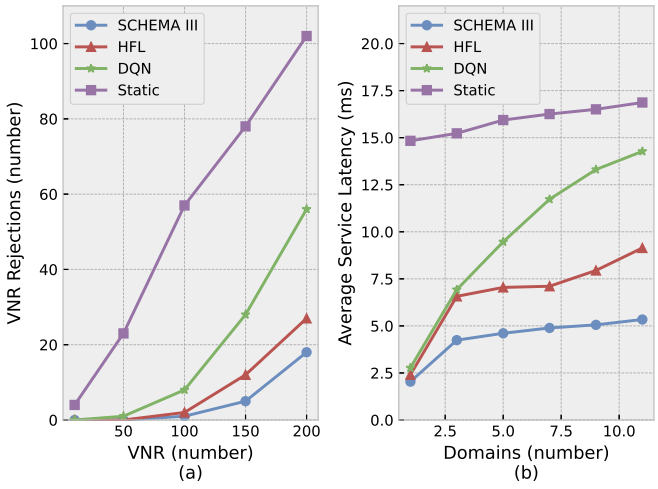

Figure 4. (a) Number of VNR rejections compared to the number Users. (b) Average perceived latency per number of autonomous network domains. Lower is better for both figures.

In Fig. 4a, we depict the number of rejected VNRs per total number of VNRs. We assume that the arrival of VNRs is subject to a Poisson process with a fluctuating arrival rate. Each VNR is has a unique computing, networking requirements and assigned lifespan. We can see that SCHEMA III had the least rejected VNRs overall. Especially, compared to the Centralized DQN solution, SCHEMA III was rejected less VNRs by 193.103% in the case of 100 users, demonstrating more successful VNR deployments in the same infrastructure. Fig. 4b is a scalability analysis of the proposed SCHEMA III VNE algorithm. The introduction of more domains in the network results in a proportional increase in the complexity of the mapping and orchestration of the underlying VNFs. Introducing more domains in the network exponentially increases the complexity of the orchestration, but as depicted in Fig. 4b, SCHEMA III and HFL distributed solutions surpass the other baselines, as the problem space grew with the introduction of new domains. The performance superiority of SCHEMA III can be attributed to the distributed architecture that distributes the state space in multiple agents.

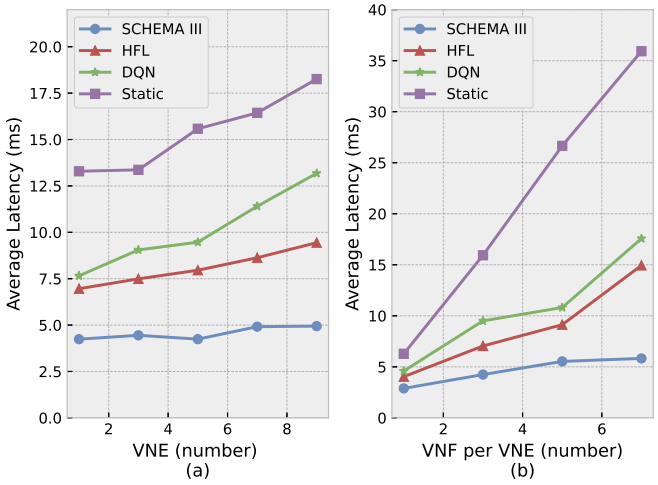

Figure 5. (a) Average slice latency per number of slices. (b) Average slice latency per number of chained slice VNFs. Lower is better for both figures.

Fig. 5a presents a comparison of all solutions regarding their ability to maintain performance during the embedding of multiple VNEs in the same network, thereby demonstrating their horizontal scalability. Consistent with the previous results, both distributed solutions have outperformed the other baseline methods. Particularly, SCHEMA III has exhibited a 114.76% difference in perceived latency compared to DQN, with 9 VNEs being embedded. The superior performance of SCHEMA III in this scenario can be attributed to its *divide et impera* approach, which effectively reduces the orchestration complexity and handles it with minimal overhead.

Fig. 5b illustrates the plot of the average latency against the number of VNFs per VNE. The findings suggest that SCHEMA III has significantly reduced latency by 144.191% compared to the Static baseline and 100.396% compared to the Centralized DQN solution, particularly in the scenario of 7 VNF per VNE. This implies that SCHEMA III has the potential to enhance the latency performance and response

times of network in scenarios involving a larger number of VNFs per VNE.

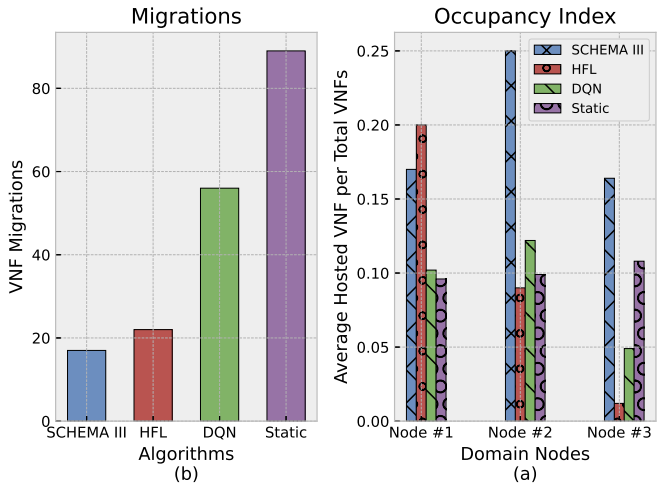

Figure 6. (a) Migration operations of 1200 simulation iterations and 5 VNE with 5 VNF each. (b) VNF occupancy index represents the average number of hosted VNFs per total number of VNFs for 1200 iterations of simulation.

Finally, Fig. 6a presents the number of VNF migration needed during 1200 simulated iterations. It is evident that the presented local RL agent, with the introduction of the Voting System, is maintaining the same VNF placement to prevent VNE re-configurations.

Fig. 6b, the behavior of SCHEMA III is illustrated, which tends to consolidate the VNFs to minimize the number of hops to the end-user and, consequently, reduce the average latency.

VI. Conclusions

The main proposition of our paper is to tackle the challenge of efficiently embedding VNs into physical networks, known as the VNE problem, by leveraging MARL algorithm. To achieve this, we propose a dynamic VNE orchestration framework based on Distributed RL that provides a scalable solution for URLLC services in multi-domain networks. Our framework consists of multiple distributed RL agents that work together to devise temporally dynamic VNE placements. The evaluation and comparison of our proposed algorithm with existing solutions in the state of the art demonstrate the effectiveness and potential of our approach. Finally, we conclude with a low latency service demonstration KPI, showcasing the practical applications of our framework. Our contributions provide a significant step towards realizing Network Virtualization (NV) in a sustainable way, addressing an important challenge in the deployment and allocation of the physical infrastructure for 5G/6G communication systems.

Acknowledgment

This work was supported by the European Commission research projects MARSAL (101017171) and 5G-ERA (101016681).

References

[1] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[2] G. Angelopoulos, A. Rossi, G. L'Arco, and S. Rossi, "Transparent interactive reinforcement learning using emotional behaviours," in *Social Robotics*, F. Cavallo, J.-J. Cabibihan, L. Fiorini, A. Sorrentino, H. He, X. Liu, Y. Matsumoto, and S. S. Ge, Eds. Cham: Springer Nature Switzerland, 2022, pp. 300–311.

[3] H. Yao, S. Ma, J. Wang, P. Zhang, C. Jiang, and S. Guo, "A continuous-decision virtual network embedding scheme relying on reinforcement learning," *IEEE Transactions on Network and Service Management*, vol. 17, no. 2, pp. 864–875, 2020.

[4] Z. Yan, J. Ge, Y. Wu, L. Li, and T. Li, "Automatic virtual network embedding: A deep reinforcement learning approach with graph convolutional networks," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 6, pp. 1040–1057, 2020.

[5] W. Chen, X. Qiu, T. Cai, H. N. Dai, Z. Zheng, and Y. Zhang, "Deep reinforcement learning for internet of things: A comprehensive survey," *IEEE Communications Surveys Tutorials*, vol. 23, no. 3, pp. 1659–1692, 2021.

[6] P. Zhang, N. Chen, S. Li, K.-K. R. Choo, and C. Jiang, "Multi-domain virtual network embedding algorithm based on horizontal federated learning," *IEEE Access*, vol. 9, pp. 104 610–104 620, 2021.

[7] A. Dalgkitsis, P.-V. Mekikis, A. Antonopoulos, G. Kormentzas, and C. Verikoukis, "Dynamic resource aware vnf placement with deep reinforcement learning for 5g networks," in *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, 2020, pp. 1–6.

[8] A. Dalgkitsis, L. Garrido, K. Ramantas, L. Alonso, and C. Verikoukis, "Schema: Service chain elastic management with distributed reinforcement learning," in *GLOBECOM 2021 - 2021 IEEE Global Communications Conference*, 2021, pp. 1–6.

[9] A. Dalgkitsis, L. A. Garrido, F. Rezazadeh, H. Chergui, K. Ramantas, J. S. Vardakas, and C. Verikoukis, "Sche2ma: Scalable, energy-aware, multidomain orchestration for beyond-5g urllc services," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–11, 2022.

[10] A. Dalgkitsis, A. Chawla, A.-M. Bosneag, and C. Verikoukis, "Safeschema: Multi-domain orchestration of slices based on saferl for b5g networks," in *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*, 2022, pp. 3435–3440.

[11] M. Abadi *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[12] F. Chollet *et al.*, "Keras," https://keras.io, 2015.

[13] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," 2016.

[14] S. Knight, H. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The internet topology zoo," *Selected Areas in Communications, IEEE Journal on*, vol. 29, no. 9, pp. 1765 –1775, october 2011.

[15] J. Li, W. Shi, H. Wu, S. Zhang, and X. Shen, "Cost-aware dynamic sfc mapping and scheduling in sdn/nfv-enabled space–air–ground-integrated networks for internet of vehicles," *IEEE Internet of Things Journal*, vol. 9, no. 8, pp. 5824–5838, 2022.

[16] I. Maity, T. X. Vu, S. Chatzinotas, and M. Minardi, "D-vine: Dynamic virtual network embedding in non-terrestrial networks," in *2022 IEEE Wireless Communications and Networking Conference (WCNC)*, 2022, pp. 166–171.