Energy Efficient VNF-FG Embedding via Attention-based Deep Reinforcement Learning

Omar Houidi*, Oussama Soualah†, Ines Houidi‡, Djamal Zeghlache*
*Telecom SudParis, SAMOVAR, Institut Polytechnique de Paris, France
†OS-Consulting, 79 avenue François Mitterrand, Athis Mons, France
‡ReDCAD Lab, National Engineering School of Sfax, University of Sfax, Tunisia
Email: {omar.houidi, djamal.zeghlache}@telecom-sudparis.eu, oussama.soualah@os-c.fr, ines.houidi@enis.tn

***Abstract*—Designing smart mechanisms to facilitate and accelerate service deployment and management is one of the most challenging aspects for network infrastructure providers. This is due to the massive amount of traffic that they are expected to support, the decentralized nature of the architectures, and the services they run to meet quality targets and avoid Service Level Agreement (SLA) violations. Therefore, Communications Service Providers (CSPs) are devoting much of their efforts on reducing energy consumption and reducing carbon footprint of their network infrastructures. In future communication networks, traditional management mechanisms, and centralized legacy solutions show their limitations in ensuring revenue for the infrastructure providers, the service providers, and a good Quality of Experience (QoE) for the end-users. The deployment of these services requires, typically, an efficient allocation of Virtual Network Function Forwarding Graph (VNF-FG). In this context, we propose an intelligent energy efficient VNF-FG embedding approach based on multi-agent attention-based Deep Reinforcement Learning (DRL). Our contribution uses a semi-distributed DRL mechanism for VNF-FG placement. The proposed algorithm is shown to outperform previous state-of-the-art approaches in terms of acceptance rate, power consumption, and execution time.**

***Index Terms*—Energy efficiency, Deep Reinforcement Learning, Attention, Multi-Agent, Virtual Network Function Embedding.**

I. Introduction

The widespread interconnection of millions of devices increases energy consumption and thus, carbon emissions. Furthermore, factors such as the production of billions of devices, their shipment, and the excessive use of data centers and network resources, all contribute to an increase in power consumption. In this context, the role of network management and control technology is to contribute to energy savings. Boosting transmit power between the increasing number of connected devices in order to increase communication capacity may result in expensive operating costs. Because energy is a crucial OPerating EXpenditure (OPEX) factor, its smart control is regarded as necessary for network expansion and function. Network providers face the challenge of power supply to their large-scale networks.

Network Function Virtualization (NFV) [1] decouples network functions from dedicated hardware devices (the traditional middleboxes). This decoupling enables the hosting of network services, known as Virtualized Network Functions (VNFs), on commodity hardware (such as servers), facilitates and accelerates service deployment and management by providers, improves flexibility, leads to efficient and scalable resource usage, and reduces costs.

For network management in NFV environments, researchers emphasize the importance of AI-native network slicing for VNF-FG embedding. Reinforcement Learning (RL) approaches can be used to improve decision-making speed and accuracy when dealing with large requests for dynamically constructing and updating end-to-end (E2E) slices (or VNF-FG) across multiple infrastructures. The development of an efficient distributed machine learning-based handover decision algorithm aids in the resolution of energy savings and scalability issues. A holistic intelligent management and orchestration system with self-diagnosis, self-healing, and self-configuration is required to reduce operational costs and complexity.

A widely used Multi-Agent Reinforcement Learning (MARL) paradigm known as *Centralized Training and Decentralized Execution* (CTDE) [2], learns agent policies centrally and executes the derived policies decentrally. Several CTDE learning approaches have been proposed, including both policy gradient and value-based methods [3] shown to achieve reasonably good performance in challenging tasks. Despite its success, the execution of fully decentralized policies suffers from limitations, particularly when agents have partial observability in a stochastic environment. In fact, during decentralized sequential execution, an agent's uncertainty about the states and actions of other agents can be aggravated and lead to suboptimal policies.

Motivated by these observations, in this paper, we design MA3C, a novel deep MARL architecture with a semi-distributed fashion, that can significantly improve energy efficiency by using a *Multi-Agent Actor-Attention-Critic* mechanism for smart cooperation between agents without degrading accuracy and performance. MA3C combines multi-head attention [4] and makes decisions on VNF-FG embedding by learning a centralized joint action-value function and by using it to guide the optimization of decentralized policies at each agent. Compared to the existing methods, the main contributions of this paper can be summarized as follows:

- We formulate the VNF-FG embedding problem as an optimization model and establish a reward function aiming to solve a trade-off between maximizing acceptance

978-3-903176-59-1 ©2023 IFIP

rate (and thus the providers' revenue) and minimizing the power consumption.
- We propose a novel attention mechanism-based Deep Deterministic Policy Gradients (DDPG) framework, using the Actor-Critic network structure.
- Our model takes into consideration the features of physical hosts and links and outputs the mapping decision of VNFs on physical hosts and steering inter-VNF traffic across the hosts with verification of constraints.
- Through extensive simulation experiments, we show that our MA3C method outperforms the state-of-the-art algorithms in terms of acceptance ratio, power consumption, and execution time.

The remainder of this paper is organized as follows. Section II describes related work and provides an overview of existing works on energy-efficient VNF forwarding and MARL methods. Section III formulates the VNF-FG embedding problem and Section IV describes the proposed model. Section VI reports performance evaluation and simulation results, and Section VII summarizes the main findings.

II. Related Work

The need to dynamically deploy virtual network services on-demand, through VNF-FG embedding, is identified as a core technology of 5G/6G networks. Therefore, this issue has been at the very center of academic and industrial research in recent years. As comprehensive surveys are already given in [1] and [5], we only give a short summary of VNF-FG embedding related works, as well as the use of RL in networking and especially in the VNF-FG placement and chaining domain.

A. Combinatorial optimization theory for NFV

VNF-FG embedding is formulated as an Integer Linear Programming (ILP) in [6]–[8] to find exact solutions for hosting the VNFs of the requested service graph. As the addressed problem is NP-Hard, the exact solutions do not scale with size and require an excessive amount of time to find the optimal solutions. Heuristic algorithms [9]–[11] are typically and consequently proposed to scale better with problem size by solving the problem iteratively and to find good solutions much faster. Unfortunately, this is accomplished at the expense of quality of the solutions. Note that the above works based on combinatorial optimization theory cannot cope with real-time dynamic network variations [12].

B. Deep Reinforcement Learning for NFV

Since VNF-FG embedding problem can be well described as a Markov Decision Process (MDP), then Reinforcement Learning (RL) (more precisely, Deep RL (DRL)) is an appropriate framework to use to find approximate optimal solutions and realize long-term reward. The DRL agent learns by interacting with the environment. Using the experience gathered, the DRL agent is able to optimize some objectives given in the form of cumulative rewards. Some recent works [12]–[14] propose DRL-based approaches for VNF-FG embedding.

However, none of the aforementioned works take the impact of surrounding nodes' resources on network states into account. In fact, the DRL model distinguishes the importance of neighbors to the learning agent based on their remaining resources. The attention mechanism allows for the selection of neighbor nodes with sufficient resources and contributes to the generation of neighbor interaction behaviors.

C. Investigations on Energy-Awareness VNF-FG placement methods

VNF-FG deployment has been optimized through various objectives and aspects. However, the primary consideration of energy remains an open challenging topic, particularly in a graph-structured Service Function Chain (SFC) environment. In [15], authors modeled the energy-efficient graph-structured SFC as a combinatorial optimization problem, where a Graph Convolutional Network GCN-based DRL was proposed to minimize the energy consumption and autonomously select nodes with adequate resources. It is worth noting that [16] is one among few attempts to address the problem of service function chaining with the goal of minimizing energy consumption. Several heuristics were proposed consisting in iteratively placing the VNFs in series using the nearest search procedure. In [17], a sampling-based Markov approximation (MA) approach, using replications, is proposed for the energy-efficient VNF placement problem. This method needs a long time to find a near-optimal solution which makes it unpractical. In fact, the solutions which consider multiple parameters as in [16], [17] seem to have limited applicability and are more efficient in cases of small numbers of user nodes, due to their complexity costs. Heuristics are a good alternative in systems where the context is not very changing. In systems where constraints and objectives are changing, these types of approaches are not very suitable since they generally require a total redesign of the heuristic. Moreover, with heuristics, we have a rapid convergence at the price of the risk of sticking to a local minimum that may not be effective.

D. Our Contribution

To deal with the high-dimensional and time-varying network state, as well as the complex network environment, we are motivated to use the Deep Deterministic Policy Gradient (DDPG) [18] algorithm. A DRL agent will not typically pay equal attention to all available placement nodes. The agent usually chooses the current action based on higher levels of cognitive skill information and ignores other perceptible information. We introduce the concept of attention mechanism to simulate the agent's action for DDPG. We discover that during the DDPG training process, the attention mechanism will automatically focus on the most likely neighbor node, which may influence the agent's selection behavior. It ultimately helps to reduce attention to other unnecessary nodes and improve the model's training efficiency. With this motivation, in this paper, we propose an intelligent energy-efficient approach to address VNF-FG placement and chaining with VNFs shared across multiple tenants to optimize resource usage, reduce energy

consumption, and increase provider revenue. Our proposed algorithm MA3C solves the VNF-FG embedding problem by applying an attention mechanism combined with an Actor-Critic DRL architecture.

III. Problem Formulation

The VNF-FG placement and chaining optimization problem, extensively addressed in the literature, corresponds to the placement of the requested VNFs and flow paths in the hosting infrastructure.

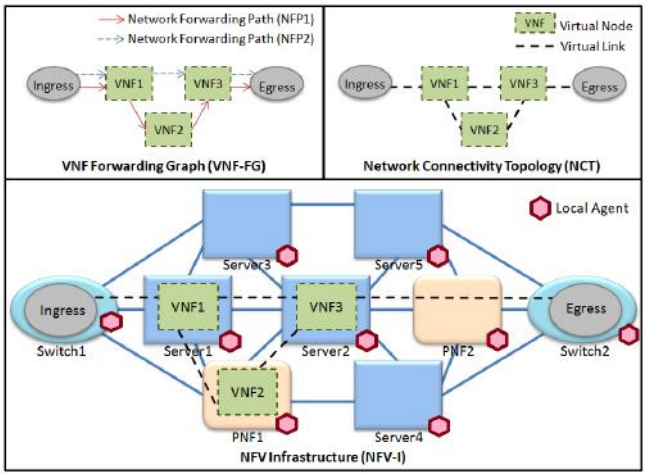

Fig. 1: Mapping of the VNF-FG in the NFV-I.

A. Substrate Graph or NFV Infrastructure Model

The physical infrastructure network is modeled as an undirected weighted graph $G_I = (N_I, E_I)$ where N_I is the set of physical network nodes and E_I is the set of physical communication links. Each physical node, $n_u^I \in N_I$, is characterized by its available processing capacity (i.e., CPU) denoted by C_u^I which represents the physical resources of the node (such as computing resources), and its type T_u^I: switch, server or Physical Network Function (PNF). The PNFs are the traditional physical middleboxes implementing network functions. An example of such an infrastructure, known as the NFV-I, including two PNFs, two switches, and some interconnected servers is presented in Figure 1. Physical communication links are represented by E_I, where $e_{uv}^I \in E_I$ is the physical link connecting two physical nodes n_u^I and n_v^I. For each physical link $e_{uv}^I \in E_I$, we denote by $B_{e_{uv}}^I$ the available bandwidth capacity of the link, and $L_{e_{uv}}^I$ the transmission delay of the link.

The power consumption, at time t, of a physical node/machine $n^I \in N_I$ is estimated as defined in [19] based on the following equation:

$$P_t(n^I) = P^i(n^I) + \left(P^M(n^I) - P^i(n^I)\right) \times U_t(n^I) \quad (1)$$

where $P^i(n^I)$ is the power consumption where the machine n^I is at the idle state. $P^M(n^I)$ is the maximum power consumption. It is reached when the physical machine is fully used. $U_t(n^I)$ is the usage rate (i.e., a value between 0 and 1) of the initial CPU capacity.

B. VNF Forwarding Graph (VNF-FG) or SFC Graph Model

The client request (i.e., a requested Service Function Chain (SFC)), is modeled as a directed graph $G_v = (N_v, E_v)$ where N_v is the set of virtual nodes and E_v is the set of virtual links in the requested graph. Each virtual node, $n_i^v \in N_v$, is characterized by its required processing capacity c_i^v and its type t_i^v: VNF or switch (i.e., ingress or egress). Each virtual link $e_{ij}^v \in E_v$ is described by its required bandwidth $b_{e_{ij}}^v$. Note that we can also consider instead the end-to-end latency if this is the desired criterion to be taken into account. Our model is generic and can be adjusted based on the client's criteria and requirements. We associate a VNF-type to each VNF to represent the network service or function type (e.g., firewall, Network Address Translation (NAT), Deep Packet Inspection (DPI), etc.). The VNFs can be hosted only by servers or PNFs having the same type. The ingress and egress nodes can be hosted only by switches. Figure 1 depicts two Network Forwarding Paths (NFP) or chains. Each Forwarding Path NFP describes the ordered VNF sequence the traffic must pass through. From the VNF forwarding graph G_v, we derive an intermediate request graph called the Network Connectivity Topology graph NCT_v. The $NCT_v = (N_v, E_v)$ is a weighted undirected graph having exactly the same set of nodes and edges as G_v. The key attribute of an NCT node $n_i^v \in N_v$ is its requested processing capacity c_i^v. The weight (or demanded bandwidth $b_{e_{ij}}^v$) of an NCT virtual link $e_{ij}^v \in E_v$ is the sum of the requested bandwidths of all the VNF flows passing through it.

C. VNF-FG Embedding Problem

The VNF-FG (Virtual Network Function-Forwarding Graph) Embedding [20] is a problem in the field of computer networking. It involves finding an optimal way to map (i.e., place and chain) the virtual network functions (VNFs) of a network onto physical network resources, such as servers, switches, and storage devices, while satisfying various constraints such as resource utilization, network bandwidth, and security. The goal of VNF-FG placement and chaining is to ensure that the VNFs (considered as nodes in a flow graph) are placed in a way that minimizes the use of resources and meets the performance requirements of the network. This is important because virtual networks can become congested and slow down if the VNFs are not placed in an optimal manner.

Figure 1 depicts a placement/embedding solution for the VNF-FG highlighted by the dashed lines, starting from the ingress switch 1, crossing the VNF 1, VNF 2, and VNF 3 (hosted respectively in Server 1, PNF 1, and Server 2), and ending at the egress switch 2. The Agent located at each physical node will have access to local information from the neighbors to make a better decision. Our case is a dynamic scenario where we do not know in advance the incoming flows.

Based on various state-of-the-art approaches, the infrastructure providers can map the VNFs using multiple objectives: such as minimizing mapping costs [12], improving energy efficiency [15], [21], maximizing acceptance ratio [22], [23], and improving provider gains [24], [25]. In this paper, our

objective is to jointly minimize the power consumption of NFV providers and maximize the acceptance rate of requests.

IV. Multi-Agent Attention Actor-Critic (MA3C)

The main idea behind our multi-agent RL approach is to learn the critic for each agent by selectively paying attention to information from other neighboring agents. This is the same paradigm of training critics centrally and executing learned policies decentrally.

1) Attention mechanism: The attention mechanism functions in a manner similar to the differentiable key-value memory model [26]. Each agent intuitively queries the other agents for information about their observations and actions and incorporates that information into the estimation of its value-function. To calculate the Q-value function $Q_i^{\psi}(o,a)$ for the agent i, the critic receives the observations, $o = (o_1, ..., o_N)$, and actions, $a = (a_1, ..., a_N)$, for all agents indexed by $i \in \{1...N\}$.

$Q_i^{\psi}(o,a)$ is a function of the agent i's observation and action, as well as other agents' contributions:

$$Q_i^{\psi}(o,a) = f_i(g_i(o_i,a_i), x_i) \tag{2}$$

where f_i is a two-layer multi-layer perceptron (MLP), while g_i is a one-layer MLP embedding function. The contribution from the other agents, x_i, is a weighted sum of each agent's value:

$$x_i = \sum_{j \neq i} \alpha_j v_j = \sum_{j \neq i} \alpha_j h(V g_j(o_j, a_j)) \tag{3}$$

where the value, v_j is a function of agent j's embedding, encoded with an embedding function and then linearly transformed by a shared matrix V. Note that h is an element-wise non-linearity (Leaky Rectified Linear Unit, or Leaky ReLU was used in our case).

The attention weight α_j compares the embedding e_j with $e_i = g_i(o_i, a_i)$, using a bilinear mapping (i.e., the query-key system) and passes the similarity value between these two embeddings into a *softmax* function.

$$\alpha_j = \exp(W_q e_i (W_k e_j)^T) \tag{4}$$

where W_q transforms e_i into a "query" and W_k transforms e_j into a "key". The matching is then scaled by the dimensionality of these two matrices to prevent vanishing the gradients [4]. In our experiments, we have used multiple attention heads [4]. In this case, each head, using a separate set of parameters (W_k, W_q, V), generates an aggregated contribution from all other agents to the agent i and we simply concatenate the contributions from all heads as a single vector. Importantly, each head can concentrate on a different weighted mixture of agents.

Note that the weights for extracting selectors (i.e., queries), keys, and values are shared across all agents, which encourages a common embedding space. Because multi-agent value-function approximation is essentially a multi-task regression problem, the critic parameters can be shared between agents even in adversarial settings. This parameter sharing allows our method to learn effectively in environments where local rewards for individual agents are different but share common features. This method can easily be extended to include additional information, beyond local observations and actions, at training time, including the global state if it is available, simply by adding additional encoders as in [27]. In fact, we do not consider this case in our experiments, however, as our approach is effective in combining local observations (i.e., received from the neighboring agents) to predict expected returns in environments where the global state may not be available (the idea is to avoid using a fixed adjacency matrix that represents the global view of the network topology graph (NFV-I) indicating the set of agents, as well as the set of direct links between them).

2) Learning with Attentive Critics: All critics are updated together to minimize a joint regression loss function, due to the parameter sharing:

$$\mathcal{L}_Q(\psi) = \sum_{i=1}^{N} \mathbb{E}_{(o,a,r,o') \sim D} \left[(Q_i^{\psi}(o,a) - y_i)^2 \right], \tag{5}$$

where $y_i = r_i + \gamma \mathbb{E}_{a' \sim \pi_{\bar{\theta}}(o')} \left[Q_i^{\bar{\psi}}(o', a') - \alpha \log(\pi_{\bar{\theta}_i}(a'_i | o'_i)) \right]$

$\bar{\psi}$ and $\bar{\theta}$ are the parameters of the target critics and target policies respectively. Note that Q_i^{ψ}, the action-value estimation for agent i, receives observations and actions for all agents. α is the temperature parameter determining the balance between maximizing entropy and rewards. The individual policies are updated by ascent with the gradient technique that aims to estimate the gradient of an agent's expected returns with respect to the parameters of its policy.

Note that we are sampling all actions a, from all agents' current policies in order to calculate the gradient estimate for agent i, unlike in the MADDPG algorithm [28], where the other agents' actions are sampled from the replay buffer, potentially causing over-generalization where agents fail to coordinate based on their current policies.

3) Semi-distributed Actor-Critic Architecture: This section presents the actor-critic architecture used for our semi-distributed model. Our setting combines global critics $Qg_{i,j}$ with a set of agents $A_{i,j}$ distributed inside the N nodes. In particular, a critic (or a number of critics) is centrally learned with information from all agents. As the input of the critic is the distribution of actions and the state of the network, it should be able to model with precision the future behavior (and so the future reward) of the network. The global/discounted reward R can be expressed as the average combination of local rewards $r_{i,j}$ ($R = \frac{1}{N^2} \sum_{i,j} r_{i,j}$).

V. States, Actions, and Rewards

States: The traffic matrix has been used as the system's states in routing problems. In fact, the traffic matrix can be seen as the client's requests to the physical networks. We adopt a similar idea to the VNF-FG embedding problem where the client requests are represented in the form of VNF-FGs. Therefore, the descriptions of VNF-FGs can be formulated as the states as in [29]. A VNF-FG can be expressed as

the chain of VNFs where each VNF has specific resource requirements. Besides VNFs, Virtual Links (VLs) have specific Quality of Service (QoS) requirements, (such as latency, end-to-end delay, etc). The VNF-FGs can be described as a vector that expresses the demanded computing resources of VNFs, as well as the QoS requirements of VLs. The DRL agent receives this vector and analyzes the remaining resources of the physical nodes and links in order to select an action that reflects the VNF-FGs mapping into the physical networks.

Actions: Our action represents the mapping of VNF-FG graphs into the substrate networks. We introduce auxiliary variables $a_t^{n^I,n^v} \in [0,1]$ that indicate the priority of assigning VNF n^v at substrate node n^I in time-step t. Furthermore, we introduce the auxiliary variables $w_t^{e^I,e^v}$ as the weights of links which will be exploited by Dijkstra's algorithm based on available bandwidth to find the path for VL e^v at time-step t. In fact, the Dijkstra algorithm takes into account all weights to identify the substrate candidate path for each virtual link.

Rewards: We adopt both the acceptance rate and the power consumption as the reward for an action. A VNF-FG is deployed when all VNFs and VLs are deployed successfully. The requests of VNFs and VLs at time-step t are described by s_t and the allocation of VNFs and VLs in the substrate network is described by a_t. The acceptance ratio (AR) has recently been adopted to assess the performance of VNF-FG embedding algorithms [30]. The reward at time-step t, (r_t) depends on the requests states (s_t) and allocation action (a_t). Unlike previous approaches, which just use the acceptance ratio as a simple reward function, we introduce in our proposed algorithm MA3C, as shown in Equation 6, a constrained reward expression to penalize decisions if the consumed power at time t, $P_G(t)$ violates a tolerable bound, called the *Reference Power*. Equation 6 presents the reward expression used for comparisons in the performance evaluation.

$$\mathcal{R} = AR - \lambda \times c(s,a) \tag{6}$$

where $c(s,a) = |Reference_{Power} - Current_{Power}(s,a)|$ is the penalty term. The constrained reward uses parameter λ to penalize actions violating the power consumption limit. In our case, we implement the RCPO [31] algorithm, which includes the SLA requirements and constraints directly in the optimization (i.e., in the modeling framework), to find optimal λ values (i.e., Lagrangian multipliers). In fact, we adopt the same approach as RCPO and embed in the reward function penalty when the power consumption target is violated or not met to provide performance guarantees.

VI. Performance Evaluation

This section presents the evaluation methodology and the simulation results. The algorithms are compared using extensive simulations where both the requests and hosting infrastructures are drawn using standard graph generation tools (such as the GT-ITM Tool) and real infrastructure topologies (such as the Germany50 network topology [32]). We evaluate the performance of our proposed algorithm and compare it with **NFVdeep** [12]: a state-of-the-art approach for VNF-FG Placement and Chaining problem. NFVdeep method is a type of RL technique that relies upon optimizing parameterized policies concerning the expected return (long-term cumulative reward) by gradient descent, and **EE-TCA** [33]: an Energy Efficient Tree search-based Chain placement Algorithm that is an extension of the Monte Carlo Tree Search (MCTS) method.

Our Multi-Agent Attention Actor-Critic (MA3C) solution was evaluated using an experimental platform where the algorithms have been deployed and evaluated at small scale. Then, the second evaluation relies on a simulation of the requests and the infrastructure at much larger scale to complete the performance analysis (i.e., our approach was tested against a wide variety of different topologies to show that the proposed solution works independently of a fixed topology).

A. Evaluation using an experimental platform

To conduct experiments in a real cloud environment, we use 50 servers from the Network and Cloud Federation (NCF) platform, which constitute our dedicated private infrastructure [34]. Our proposed algorithms are integrated in the smart placement module, acting as one key component of the cloud framework, responsible for VNF-FGs embedding.

The Germany50 network topology [32] was used for the first assessment (with 50 nodes). The capacities of physical nodes and links are generated randomly in the [100, 120] interval. The size of the VNF-FG requests is arbitrarily set to 5 nodes. The requested CPU of each VNF depends on its type: Firewall (4 CPUs), Proxy (2 CPUs), NAT (1 CPUs), and IDS (8 CPUs). We generate 1000 VNF-FG requests with various VNFs combinations selected randomly. The lifetime of each request follows an exponential distribution with a mean of 1000 time units. The VNF-FG requests are generated using a Poisson process with an average arrival rate λ of 5 requests per 100 time units. We experimentally evaluate the acceptance rate of the incoming requests, and the energy consumption of our algorithm. We extended the service lifecycle manager of our experimental framework [34] by integrating MA3C our new smart energy-efficient VNF-FG placement module coupled with a PyTorch environment for the machine learning framework where the DRL actor-critic based approaches have been implemented and used to conduct the evaluation. This framework is used as a benchmarking environment for smart placement algorithms.

B. Metrics and performance indicators

1) ***Acceptance Rate****:* is the acceptance rate of the incoming requests during the simulation. In other terms, the rate of VNF chain requests that have been accepted due to the physical resource shortage (i.e., available CPU).

2) ***Power Consumption****:* The global consumed power at time t, $P_G(t)$, is equal to the sum of the power consumption of the activated servers. Formally, $P_G(t) = \sum_{n^I \in N_I} P_t(n^I)$.

3) ***Execution time****:* is the time needed to find an embedding solution for one VNF-FG request. This metric reflects the ability of the algorithms to scale with problem size.

TABLE I: MA3C parameters

Actor learning rate	10^{-3}
Critic learning rate	10^{-2}
Batch size	32
τ target network update rate	0.1
γ discount factor	0.9
Number of hidden layers	2
Dimension of hidden layers	64
Temperature parameter used for entropy-based exploration	10^{-3}
Policy refreshing period (time period)	$10s$

The hyper-parameters used for the performance evaluation of our algorithm are specified in Table I.

C. Evaluation Results

The performance of the algorithms is reported for the Germany50 network topology for small scale problems (50 nodes) and for randomly generated network topologies for larger problems involving hundreds of nodes and links.

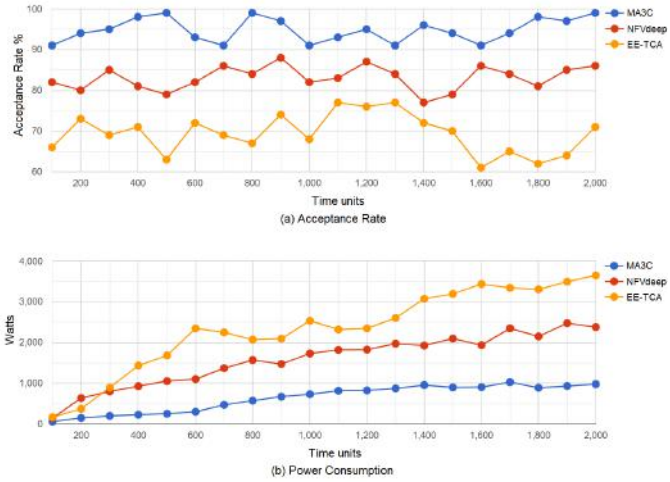

Fig. 2: Germany50 network topology results.

*1) **Realistic topology evaluation**:* From the results collected on the experimental platform using 50 servers, reported in Figure 2, we see that our proposed algorithm outperforms all other algorithms in all the assessed performance metrics (power consumption, and acceptance rate). For the power consumption metric, MA3C consumes less than half of the power used by the other algorithms over all the evaluation period (i.e., more precisely, less than 1000 watts). In terms of acceptance rate, the same behavior can be observed where our MA3C algorithm outperforms the NFVdeep and EE-TCA based solutions by accepting more requests (more than 90%) in short and long-term.

*2) **Large-scale evaluation**:* To analyze the behavior of the algorithms and their scalability with problem size, larger hosting infrastructures (NFV-Is with 200 nodes) are randomly generated (using the GT-ITM Tool) and used in this second performance assessment. This evaluation should confirm the previous results and reveal the ability of the algorithms to scale with increasing problem size. In addition to the previous performance metrics, we report the execution time of each algorithm to shed light on their scalability.

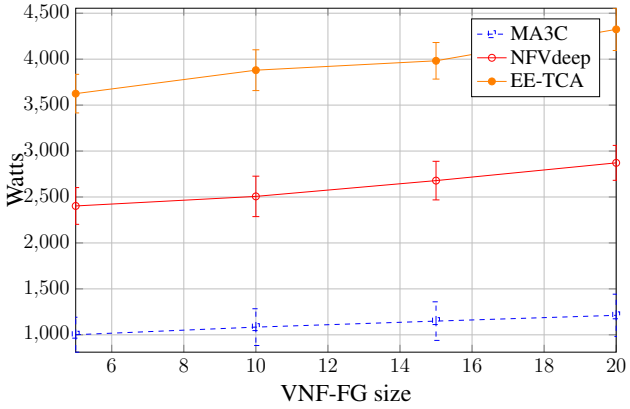

Fig. 3: Power consumption w.r.t VNF-FG size variation (NFV-I size = 200, VNF-FG size = up to 20).

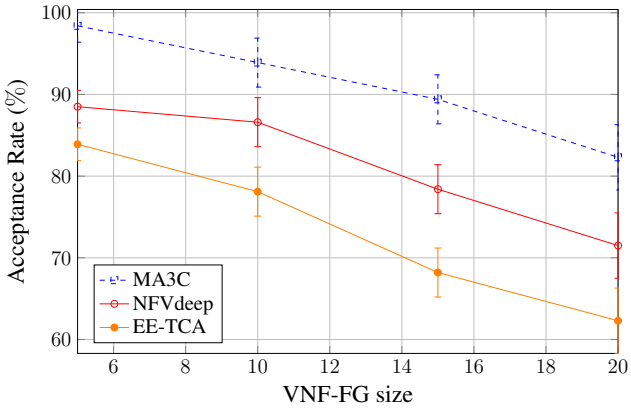

Fig. 4: Acceptance percentage w.r.t VNF-FG size variation (NFV-I size = 200, VNF-FG size = up to 20).

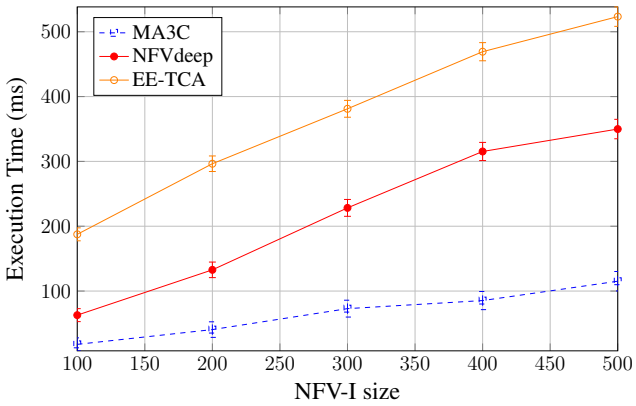

Fig. 5: Execution Time w.r.t NFV-I size variation (NFV-I size = up to 500, VNF-FG size = 5).

Figure 3 and 4 report the power consumption and average acceptance percentage of the algorithms as a function of increasing VNF-FG request sizes (from 5 to 20) and confirm the previous findings. Our MA3C algorithm outperforms the

NFVdeep and EE-TCA based solutions by accepting more requests and consuming less than half of the power used by the other algorithms over all the evaluation periods and scenarios. At high load (case where VNF-FG size of 20 nodes), MA3C is also less power consuming and can even reduce power consumption in extreme saturation conditions thanks to consolidation and smart inter-agent cooperation. The results in Figure 3 and 4, presented with a 95% confidence interval, depict a high similarity with results in Figure 2(a) and 2(b) (more precisely for VNF-FG size of 5 nodes).

*3) **Execution time (Convergence time)**:* Figure 5 reports the execution time performance of the algorithms with problem size to gain insight on their scalability and complexity. We generate 1000 VNF-FG requests and vary the substrate graph sizes from 100 to 500 nodes. The MA3C-based algorithm has significantly better execution time compared with the NFVdeep and EE-TCA execution times.

VII. Conclusion

This paper proposes a smart energy-efficient algorithm for VNF-FG embedding in NFV-enabled infrastructures. Using an attention-based multi-agent DRL mechanism, our solution reduces the global data center power consumption through consolidation at the hardware level and optimizes the QoE/QoS for the clients by optimally sharing the VNFs across multiple tenants at the service level. Performance evaluation using a real testbed cloud environment confirms the energy efficiency of our MA3C algorithm. Simulation results for large-scale topologies highlight the scalability of our proposal ensured by the semi-distributed architecture philosophy. Our approach also ensures a tradeoff between energy savings and scalability.

References

[1] B. Yi, X. Wang, K. Li, S. K. Das, and M. Huang, "A comprehensive survey of Network Function Virtualization," *Computer Networks*, vol. 133, pp. 212–262, 2018.

[2] L. Kraemer and B. Banerjee, "Multi-agent reinforcement learning as a rehearsal for decentralized planning," *Neurocomputing*, vol. 190, pp. 82–94, 2016.

[3] A. Oroojlooy and D. Hajinezhad, "A review of cooperative multi-agent deep reinforcement learning," *Applied Intelligence*, vol. 53, no. 11, pp. 13 677–13 722, 2023.

[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[5] J. Gil-Herrera and J. F. Botero, "Resource Allocation in NFV: A Comprehensive Survey," *IEEE Trans. Network and Service Management*, vol. 13, no. 3, pp. 518–532, 2016.

[6] M. C. Luizelli, L. R. Bays, L. S. Buriol, M. P. Barcellos, and L. P. Gaspary, "Piecing together the NFV provisioning puzzle: Efficient placement and chaining of virtual network functions," in *IFIP/IEEE IM*, 2015, pp. 98–106.

[7] M. F. Bari, S. R. Chowdhury, R. Ahmed, and R. Boutaba, "On orchestrating virtual network functions," in *IEEE CNSM*, 2015, pp. 50–56.

[8] H. Moens and F. D. Turck, "VNF-P: A model for efficient placement of virtualized network functions," in *IEEE CNSM*, 2014, pp. 418–423.

[9] O. Houidi, O. Soualah, W. Louati, M. Mechtri, D. Zeghlache, and F. Kamoun, "An efficient algorithm for virtual network function scaling," in *IEEE GLOBECOM*, 2017, pp. 1–7.

[10] J. Pei, P. Hong, K. Xue, and D. Li, "Efficiently embedding service function chains with dynamic virtual network function placement in geo-distributed cloud system," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 10, pp. 2179–2192, 2018.

[11] G. Sallam and B. Ji, "Joint placement and allocation of virtual network functions with budget and capacity constraints," in *IEEE INFOCOM*, 2019, pp. 523–531.

[12] 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 *Proceedings of the International Symposium on Quality of Service*, 2019, pp. 1–10.

[13] P. T. A. Quang, A. Bradai, K. D. Singh, and Y. H. Aoul, "Multi-domain non-cooperative VNF-FG embedding: A deep reinforcement learning approach," in *IEEE INFOCOM*, 2019, pp. 886–891.

[14] S. I. Kim and H. S. Kim, "A research on dynamic service function chaining based on reinforcement learning using resource usage," in *ICUFN 2017, Milan, Italy, July 4-7, 2017*, 2017, pp. 582–586.

[15] Q. Siyu, L. Shuopeng, L. Shaofu, M. Y. Saidi, and C. Ken, "Energy-efficient VNF deployment for graph-structured SFC based on graph neural network and constrained deep reinforcement learning," in *IEEE APNOMS*, 2021, pp. 348–353.

[16] M. M. Tajiki, S. Salsano, L. Chiaraviglio, M. Shojafar, and B. Akbari, "Joint energy efficient and QoS-aware path allocation and VNF placement for service function chaining," *IEEE Transactions on Network and Service Management*, vol. 16, no. 1, pp. 374–388, 2018.

[17] C. Pham, N. H. Tran, S. Ren, W. Saad, and C. S. Hong, "Traffic-aware and energy-efficient vNF placement for service chaining: Joint sampling and matching approach," *IEEE Transactions on Services Computing*, vol. 13, no. 1, pp. 172–185, 2017.

[18] I. Grondman, L. Busoniu, G. A. D. Lopes, and R. Babuska, "A Survey of Actor-Critic Reinforcement Learning: Standard and Natural Policy Gradients," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 6, pp. 1291–1307, 2012.

[19] M. Pedram and I. Hwang, "Power and performance modeling in a virtualized server system," in *2010 39th International Conference on Parallel Processing Workshops*, 2010, pp. 520–526.

[20] O. Houidi, "Algorithms for virtual network functions chaining," Ph.D. dissertation, Institut polytechnique de Paris, 2020.

[21] O. Soualah, M. Mechtri, C. Ghribi, and D. Zeghlache, "A green VNFs placement and chaining algorithm," in *IEEE/IFIP NOMS*, 2018, pp. 1–5.

[22] O. Houidi, O. Soualah, W. Louati, and D. Zeghlache, "Dynamic VNF forwarding graph extension algorithms," *IEEE Transactions on Network and Service Management*, vol. 17, no. 3, pp. 1389–1402, 2020.

[23] O. Soualah, O. Houidi, and D. Zeghlache, "A Monitoring Aware Strategy for 5G Core Slice Embedding," in *International Conference on Advanced Information Networking and Applications*. Springer, 2021, pp. 727–744.

[24] O. Houidi, O. Soualah, W. Louati, D. Zeghlache, and F. Kamoun, "Virtualized network services extension algorithms," in *IEEE NCA*, 2018, pp. 1–5.

[25] O. Houidi, O. Soualah, W. Louati, and D. Zeghlache, "An Enhanced Reinforcement Learning Approach for Dynamic Placement of Virtual Network Functions," in *IEEE PIMRC*, 2020, pp. 1–7.

[26] A. Graves, G. Wayne, and I. Danihelka, "Neural turing machines," *arXiv preprint arXiv:1410.5401*, 2014.

[27] O. Houidi, S. Bakri, and D. Zeghlache, "Multi-Agent Graph Convolutional Reinforcement Learning for Intelligent Load Balancing," in *IEEE/IFIP NOMS*, 2022, pp. 1–6.

[28] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments," in *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[29] 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.

[30] R. Riggio, A. Bradai, D. Harutyunyan, T. Rasheed, and T. Ahmed, "Scheduling wireless virtual networks functions," *IEEE Transactions on network and service management*, vol. 13, no. 2, pp. 240–252, 2016.

[31] C. Tessler, D. J. Mankowitz, and S. Mannor, "Reward Constrained Policy Optimization," *CoRR*, vol. abs/1805.11074, 2018.

[32] S. Orlowski, R. Wessäly, M. Pióro, and A. Tomaszewski, "SNDlib 1.0–Survivable Network Design Library," *Networks*, vol. 55, no. 3, pp. 276–286, 2010.

[33] O. Soualah, M. Mechtri, C. Ghribi, and D. Zeghlache, "Energy efficient algorithm for VNF placement and chaining," in *IEEE/ACM CCGRID*, 2017, pp. 579–588.

[34] M. Mechtri, C. Ghribi, O. Soualah, and D. Zeghlache, "NFV orchestration framework addressing SFC challenges," *IEEE Communications Magazine*, vol. 55, no. 6, pp. 16–23, 2017.