IEEE ComSoc | IEEE COMPUTER SOCIETY | IEEE Signal Processing Society | VTS | IEEE Transactions on Machine Learning in Communications and Networking

Received 28 February 2024; revised 10 July 2024; accepted 23 September 2024.
Date of publication 26 September 2024; date of current version 9 October 2024.

The associate editor coordinating the review of this article and approving it for publication was E. E. Tsiropoulou.

Digital Object Identifier 10.1109/TMLCN.2024.3469131

Optimizing Resource Fragmentation in Virtual Network Function Placement Using Deep Reinforcement Learning

RAMY MOHAMED[1] (Member, IEEE), MARIOS AVGERIS[1] (Member, IEEE), ARIS LEIVADEAS[2] (Senior Member, IEEE), AND IOANNIS LAMBADARIS[1] (Senior Member, IEEE)

[1]Department of Systems and Computer Engineering, Carleton University, Ottawa, ON K1S 5B6, Canada
[2]Department of Software and IT Engineering,École de technologie supérieure, Montreal, QC H3C 1K3, Canada

Corresponding author: Ramy Mohamed (ramy.mohamed@carleton.ca)

This work was supported in part by Ericsson Canada, and in part by the Natural Sciences and Engineering Research Council of Canada (NSERC) under Grant ALLRP 561415-20.

ABSTRACT In the 6G wireless era, the strategical deployment of Virtual Network Functions (VNFs) within a network infrastructure that optimizes resource utilization while fulfilling performance criteria is critical for successfully implementing the Network Function Virtualization (NFV) paradigm across the Edge-to-Cloud continuum. This is especially prominent when resource fragmentation –where available resources become isolated and underutilized– becomes an issue due to the frequent reallocations of VNFs. However, traditional optimization methods often struggle to deal with the dynamic and complex nature of the VNF placement problem when fragmentation is considered. This study proposes a novel online VNF placement approach for Edge/Cloud infrastructures that utilizes Deep Reinforcement Learning (DRL) and Reward Constrained Policy Optimization (RCPO) to address this problem. We combine DRL's adaptability with RCPO's constraint incorporation capabilities to ensure that the learned policies satisfy the performance and resource constraints while minimizing resource fragmentation. Specifically, the VNF placement problem is first formulated as an offline-constrained optimization problem, and then we devise an online solver using Neural Combinatorial Optimization (NCO). Our method incorporates a metric called Resource Fragmentation Degree (RFD) to quantify fragmentation in the network. Using this metric and RCPO, our NCO agent is trained to make intelligent placement decisions that reduce fragmentation and optimize resource utilization. An error correction heuristic complements the robustness of the proposed framework. Through extensive testing in a simulated environment, the proposed approach is shown to outperform state-of-the-art VNF placement techniques when it comes to minimizing resource fragmentation under constraint satisfaction guarantees.

INDEX TERMS Resource fragmentation, neural combinatorial optimization, reinforcement learning, service function chaining, virtual network function placement, 5G, 6G.

I. INTRODUCTION

THE dawn of 5G/6G networking era revealed the importance of Network Function Virtualization (NFV), a paradigm that enables decoupling network functions (e.g., Network Address Translation - NAT, Firewall, etc.) from purpose-specific hardware and implementing them as software instances running on conventional servers on the Cloud. NFV conceptualizes end-to-end network services as Service Function Chains (SFCs), i.e., graphs of interconnected Virtual Network Functions (VNFs) representing the requested services [1]. Within the SFC graph, vertices represent the VNFs, and edges define the virtual communication links between them. The underlying Network Function Virtualization Infrastructure (NFVI) is also modeled as a graph, where vertices represent the computing (servers) and forwarding nodes (routers), and edges the physical communication links. Leveraging the flexibility of virtualization, SFCs can be potentially deployed in the proximity of end

users, at the network Edge, shortening propagation delays and overall latency and resulting in improved QoS. Evidently, the Virtual Network Functions Chain Placement Problem (VNF-CPP) emerges from the need to deploy the SFCs while optimizing various criteria like operational costs, service delays and resource utilization under the system's constraints, such as limited resource capacities [2]. For a complete list of acronyms used in this paper, refer to Table 1.

VNF-CPP is an inherently complex and dynamic problem since it involves making decisions based on fluctuating network conditions and service requirements and has been proven to be NP-hard [3]. To enhance resource utilization efficiency and maximize the number of successful SFC deployments within the network infrastructure, many VNF placement algorithms have been introduced ranging from traditional heuristics [3], [4], [5] to modern learning-based solutions [6], [7]. These algorithms usually consider only the initial deployments of SFC requests in the infrastructure. However, in realistic scenarios, the dynamic arrival and departure of VNFs can have an adverse effect on the resource availability of the substrate network if they are not placed carefully. Unplanned VNF placement in infrastructure can lead to *resource fragmentation*, an often overlooked situation where available resources are not efficiently utilized and are dispersed in a non-optimal manner [8]. This can lead to reduced serving rates, higher operational costs, and generally increased complexity in resource management, as the infrastructure providers need to invest in more hardware or allocate additional resources to maintain Service Level Agreements (SLAs).

To mitigate this phenomenon, in this paper we propose a novel online VNF-placement framework for edge/cloud infrastructures, based on **F**ragmentation-**A**ware **N**eural **C**ombinatorial **O**ptimization with **R**eward constrained **P**olicy optimization, *FANCORP*. Our approach combines the capability of Deep Reinforcement Learning (DRL) to learn optimal policies in dynamic environments, with Reward Constrained Policy Optimization (RCPO) [9], which allows for incorporating additional constraints that are imposed by network operators and SLAs. Specifically, we employ the Lagrange relaxation technique to convert the constrained VNF-CPP optimization problem into an unconstrained one, by introducing a Lagrange multiplier for each constraint. These multipliers are then used to form the Lagrangian function, which consists of two parts: a reward based on the original objective function and a penalty signal formed by the weighted sum of constraints violation degrees. The DRL technique we implement for solving the unconstrained problem is Neural Combinatorial Optimization (NCO). NCO employs neural networks to solve combinatorial optimization problems by learning a representation of them, which is then used to make decisions or predictions about the optimal solution [10]. By combining NCO with RCPO during training, we utilize the penalty signal to direct the VNF placement policy towards feasible solutions that minimize resource fragmentation by specifically targeting the automatic update of the Lagrange multipliers. In this way, we ensure that the learned policies will satisfy the system's constraints. The main contribution of this paper is threefold:

1) We re-formulate the classic VNF-CPP problem for dynamic edge/cloud infrastructures by introducing the Resource Fragmentation Degree (RFD) metric [8] into the objective function. Solving the emerging offline Integer Linear Programming (ILP) problem results in resource-efficient placement decisions that minimize resource fragmentation under the system's constraints.
2) We transform the above ILP problem into a Constraint Markov Decision Process (CMDP) and we propose FANCORP, a novel online fragmentation-aware VNF placement framework that leverages NCO and RCPO to solve it. We combine the NCO's adaptability with the RCPO's constraint incorporation capabilities to ensure that the learned placement policies will satisfy the various performance and resource constraints. The robustness of this mechanism is enhanced by a complementary heuristic algorithm that validates the feasibility of the produced solutions and takes corrective actions in case of minor miscalculations.
3) We demonstrate the effectiveness of our proposed framework through extensive simulations and comparisons with state-of-the-art VNF placement techniques, showing that our approach yields superior performance in terms of optimizing resource fragmentation and serving rate under the given constraints. Additionally, we provide insights into the scalability and adaptability of our approach, highlighting its potential for practical deployments in real-world network scenarios.

The remainder of this paper is organized as follows: Section II highlights the related works. The system model and the fragmentation-aware VNF-CPP ILP formulation follow in Section III. The CMDP formulation as well as the NCO-RCPO-based solution are detailed in Section IV. In Section V we present the obtained results from the performance evaluation. Finally, in Section VI we summarize our conclusions and propose some future work directions.

II. RELATED WORK

A. VNF-CPP

During the last few years, VNF-CPP in edge/cloud infrastructures has been tackled in various ways. In [5], the authors propose a mechanism for placing delay-sensitive SFCs in a Mobile Edge Computing (MEC) – Cloud infrastructure. This sub-optimal but fast approach is based on the Tabu Search meta-heuristic and drives the system towards a low end-to-end communication (E2E) delay and a minimum deployment cost. A heuristic solution is proposed as well in [11], only this time for placing VNFs on mobile compute nodes (i.e., robots and drones). Here, the bin packing-inspired algorithm efficiently tackles both the radio coverage and the resource restrictions in this volatile 5G environment. Mao et al. in [12], come up with a two-part approximation algorithm

TABLE 1. List of acronyms.

Acronym	Definition
CMDP	Constrained Markov Decision Process
DRL	Deep Reinforcement Learning
E2E	End-to-End
GC	Greedy Correction
ILP	Integer Linear Programming
LSTM	Long Short-Term Memory
MADRL	Multi-Agent Deep Reinforcement Learning
MEC	Mobile Edge Computing
MLP	Multilayer Perceptron
MSE	Mean Squared Error
NCO	Neural Combinatorial Optimization
NFV	Network Function Virtualization
NFVI	Network Function Virtualization Infrastructure
QoS	Quality of Service
RCPO	Reward Constrained Policy Optimization
RFD	Resource Fragmentation Degree
RL	Reinforcement Learning
S2S	Sequence-to-Sequence
SFC	Service Function Chain
SLA	Service Level Agreement
VNE-RFD	Virtual Network Embedding with Resource Fragmentation Degree
VNF	Virtual Network Function
VNF-CPP	Virtual Network Function Chain Placement Problem

that prioritizes SFC placement at the edge in a way that the remaining VNFs need as few cloud resources as possible and the edge-cloud communication latency is minimized. On the other hand, Tran et al. in [13] break the problem down into a two-step process, which brings this work closer to a network planning solution: i) VNF placement and ii) VNF migration; for the former, a dynamic programming-based heuristic solution is utilized to place the VNFs considering the current traffic conditions. A similar solution is used for the migration, which serves as a corrective measure for when the traffic deviates significantly. Similarly, a VNF placement-migration framework is proposed in [14]. However, this time, the problem is tackled jointly as an online optimization problem. In particular, the solution aims to migrate some deployed VNFs to make room for more arriving requests to be accepted when the remaining resources in the infrastructure are low, and request rejections are observed.

In the 6G context, the need for low-latency services becomes even more prominent. To efficiently meet the emerging stringent service requirements, VNF placement and scheduling during SFC deployment needs to be carried out jointly. That is what the authors in [15] explore; after formulating VNF-CPP as an ILP problem, they suggest two efficient heuristics, namely a greedy-based one and a Tabu search-based algorithm, to jointly solve the problem in an effort to maximize the stakeholders' profits. On a similar note, Bagaa et al. in [16] investigate the orchestration of VNFs over multiple cloud domains, a common scenario in 6G deployments, and introduce three optimization solutions that consider two conflicting objectives: minimizing end-to-end delay and VNF relocations. By leveraging methods from the bargaining game theory for Pareto efficiency, they achieve near-optimal results for both objectives.

B. LEARNING-BASED SOLUTIONS

Traditional optimization methods like ILP solvers and meta-heuristic algorithms often struggle to cope with VNF-CPP's complexity, especially in highly dynamic environments [4]. Following the recent interdisciplinary trend of learning-based solutions, several recent research works have proposed deep reinforcement learning (DRL) and multi-agent deep reinforcement learning (MADRL) methods to address it.

For example, Solozabal et al. in [17] present a DRL-based method for optimizing the power consumption of the VNF Placement by extending the Neural Combinatorial Optimization (NCO) theory and combining it with heuristics. The importance of this novel mix of methods is the conclusion that highly competitive results can be achieved using relatively simple algorithms. Similarly, in [18] the authors formulate VNF-CPP as a Markov Decision Process (MDP), and a Q-Learning algorithm's variation is used to solve it. This variation takes into consideration the available computing and communication resources of the infrastructure. Their practical solution is shown to achieve near-optimal performance, close to that of the policy iteration-based algorithm. On the other hand, in [19], the authors explore the Age of Information (AoI) concept in multiple-source systems and its relation to the VNF placement. They propose a DRL-based approach that optimizes the acceptance ratio and minimizes the average AoI at the destination. Leveraging cooperation between agents, in [6] the authors propose a MADRL-based method for joint VNF placement and routing that adapts to changing network topologies and addresses multiple concurrent service requests with different delay and cost demands. This framework is shown to outperform its alternatives in terms of service cost and delay, while offering higher flexibility for personalized service requirements.

C. RESOURCE FRAGMENTATION IN VNF-CPP

In the NFVI domain multiple requests for SFCs and their VNFs can be deployed on top of a physical substrate network. Over time, with these VNFs' random arrivals and departures, the network's computing and communication resources can become isolated or dispersed. This creates a phenomenon often overlooked in the literature called *resource fragmentation* which results in inefficient infrastructure utilization [8].

Recently, there has been increasing individual research interest in addressing the issue of resource fragmentation on networking infrastructures [20]. For instance, Bari et al. [21] are some of the first ones that acknowledge resource fragmentation in VNF-CPP and define a dual cost to represent server and link resource fragmentation. Then, they incorporate it in their ILP formulation which they solve by introducing a near-optimal dynamic programming-based heuristic. However, such solutions, while innovative, lack flexibility and real-time adaptability in dynamic network environments. A different metric is introduced by the authors in [8], called Resource Fragmentation Degree (RFD), which quantifies the fragmentation of resources at substrate nodes and links,

motivated by the conception of connectivity in graph theory and shortest path models. Using this metric the problem of embedding virtual networks into the substrate network is formulated as a mixed-integer programming problem. Then, a proactive and a reactive online heuristic algorithm are suggested to solve the problem efficiently while considering resource fragmentation. Although promising, this technique does not account for the unpredictable nature of network demands, as it does not inherently learn from past experiences. On the other hand, Fu and Li in [22] resolves to a reactive-only solution to reduce fragmentation in cloud infrastructures. Specifically, an automatic VNF migration approach is devised to mitigate the load imbalance in the infrastructure, a factor identified as a critical cause of resource fragmentation. While the proposed scheme demonstrates improvements in resource utilization and load balancing, it primarily focuses on basic resource metrics, potentially overlooking more complex factors such as VNF migration overhead and predictive VNF load management, which could lead to suboptimal deployment decisions.

A different approach to reducing resource fragmentation during SFC deployment is followed in [23]; here, the authors suggest sharing the deployed VNF instances among different SFCs with the dual aim of improving the resource utilization and reducing the resource fragmentation generated by deploying many VNF instances. To this end, they introduce a weight-based VNF sharing scheduling approach that seemingly achieves fair scheduling among the SFC requests. However, the complexity introduced by the need for precise calculation and allocation of resources based on min-plus algebra might not be easily scalable in real-world scenarios with highly dynamic traffic patterns and diverse service requirements. Additionally, the requirement for accurate traffic prediction and service demand estimation to allocate weights and resources fairly could be challenging.

Similarly, Guo et al. in [24] propose a scheduling mechanism for sharing the already deployed VNF instances among different SFC requests, however utilizing dynamic weights this time in the objective model. Their heuristic approach based on Steiner Trees and Markov Decision Processes demonstrates near-optimal results in minimizing a cost that reflects internal resource fragmentation among others. However, VNF instance sharing can increase forwarding costs due to longer paths for some SFCs. This happens when SFCs are not deployed on the shortest path, potentially outweighing the cost savings from reduced VNF instance creation. Additionally, the complexity of the heuristic algorithm, although lower than MILP, may still be substantial for large-scale networks, affecting scalability and practicality.

D. DISCUSSION

The proposed FANCORP framework sets itself apart by leveraging DRL to learn optimal placement policies that dynamically adapt to network changes. This adaptability allows for continuous optimization of resource fragmentation, unlike the static or semi-dynamic approaches of prior works. By incorporating NCO with RCPO, we allow our solution to automatically learn the appropriate Lagrange multipliers. Unlike similar mechanisms where the multipliers are manually selected [17], this automated learning process driven by a penalty signal, systematically guides the placement policy towards constraint satisfaction, enhancing adaptability and ensuring efficient resource allocation.

Our mechanism is complemented by a greedy error correction heuristic that strengthens the robustness of our approach by ensuring that even when the agent's decisions are imperfect, there is a systematic method to correct them and maintain service quality. This comprehensive and responsive strategy positions our framework as an advanced solution capable of adapting to the current network state while proactively adjusting for future demands. The current work is an extension of our previous work in [25]. In particular, our extensions can be summarized as follows. Firstly, the system model has been revamped, and the ILP problem, the CMDP formulation, and the Lagrange relaxation are presented in depth. Secondly, a new objective is used for the emerging unconstrained optimization problem, and formal algorithmic descriptions are now provided for both the NCO-RCPO agent training as well as the FANCORP operation as a whole. Thirdly, the error correction mechanism has been updated with an improved strategy. Finally, we conduct additional experimentation that highlights the efficacy and efficiency of the proposed solution in various scenarios.

III. SYSTEM MODEL

A. EDGE/CLOUD INFRASTRUCTURE & SFC REQUESTS

We model the physical infrastructure as a 3-tier network, consisting of the Edge, Transport, and Cloud tiers, as shown in Fig. 1. Specifically, we define an undirected graph $\mathbb{G} = (\mathbb{H}, \mathbb{L})$, where $\mathbb{H}$ denotes the set of nodes and $\mathbb{L}$ the set of links. The nodes can be either servers $\mathbb{N} \subset \mathbb{H}$ or routers $\mathbb{W} \subset \mathbb{H}$, with $\mathbb{N} \cup \mathbb{W} = \mathbb{H}$. Servers are further divided based on their location into edge servers $\mathbb{N}_\mathbb{E} \subset \mathbb{N}$ and cloud servers $\mathbb{N}_\mathbb{C} \subset \mathbb{N}$, with $\mathbb{N}_\mathbb{E} \cup \mathbb{N}_\mathbb{C} = \mathbb{N}$. Similarly, the routers are divided into gateway routers $q \in \mathbb{W}_\mathbb{Q} \subset \mathbb{W}$, which are the network's entry points from where end-users connect to their requested SFCs, and traffic routers $o \in \mathbb{W}_\mathbb{O} \subset \mathbb{W}$ that interconnect the network tiers, with $\mathbb{W}_\mathbb{Q} \cup \mathbb{W}_\mathbb{O} = \mathbb{W}$. Finally, each server $n \in \mathbb{N}$ is attributed a vector of available resources $\mathbf{R}_n$ (i.e., CPU, memory and storage), while every physical link $(u, v) \in \mathbb{L}$ has a bandwidth capacity $B_{(u,v)}$ and a fixed propagation delay $d^{(p)}_{(u,v)}$, $u, v \in \mathbb{H}$.

We also assume a set of available VNFs $\mathbb{F}$ that are used to build the different types of requested network services offered as SFCs, represented by set $\mathbb{S}$. To accurately formulate an SFC request $\mathbf{r}_i \in \mathbb{S}$, first we define the topological structure of its interconnected VNF components, such that $\mathbf{f}_i = \{f_{i,0}, f_{i,1}, \ldots, f_{i,|\mathbf{f}_i|}\} \subseteq \mathbb{F}$, where $f_{i,0}$ must be placed on the gateway router, i.e., $q \in \mathbb{W}_\mathbb{Q}$, from which a user requesting the SFC $\mathbf{r}_i$ is connected to the network. VNFs

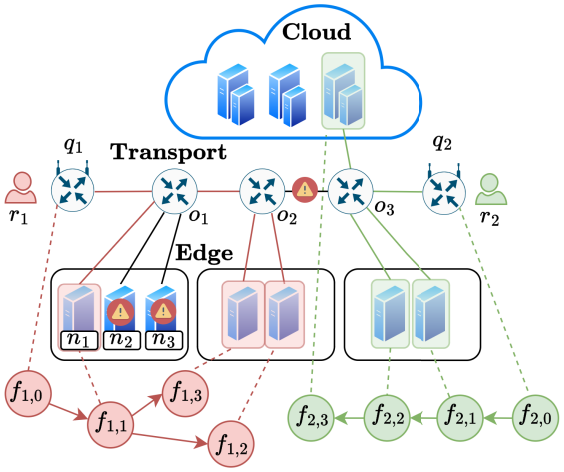


FIGURE 1. Resource fragmentation in VNF-CPP.

can be interconnected in various ways (e.g., linear or bifurcated), as explained in [26]. We define the matrix of a request's resource demands as $\boldsymbol{\sigma}_i = (\boldsymbol{\sigma}_{i,0}, \boldsymbol{\sigma}_{i,1}, \ldots, \boldsymbol{\sigma}_{i,|\mathbf{f}_i|})$, where $\boldsymbol{\sigma}_{i,k}$ is the vector of resource demands (i.e., CPU, memory and storage) of VNF $f_{i,k} \in \mathbf{f}_i$ with $k \in [0, |\mathbf{f}_i|]$. Next, we define the vector of maximum processing delays that need to be satisfied for the request's VNF components as $\mathbf{D}_i^{(c)} = (D_{i,0}^{(c)}, D_{i,1}^{(c)}, \ldots, D_{i,|\mathbf{f}_i|}^{(c)})$. We note that the incurred processing delays depend on the capabilities of the servers that will host the particular VNFs, $d_{i,k,n}^{(c)}$, $i \in |\mathbb{S}|, k \in |\mathbf{f}_i|, n \in \mathbb{N}$. Following this convention, we introduce the vectors of an SFC's bandwidth demands $\mathbf{b}_i = (b_{i,0,1}, b_{i,0,2}, \ldots, b_{i,|\mathbf{f}_i-1|,|\mathbf{f}_i|})$ and the maximum propagation delays $\mathbf{D}_i^{(p)} = (D_{i,0,1}^{(p)}, D_{i,0,2}^{(p)}, \ldots, D_{i,|\mathbf{f}_i-1|,|\mathbf{f}_i|}^{(p)})$. Finally, each SFC can tolerate a maximum end-to-end delay of $D_i^{(t)}$. Based on the above, we define an SFC request $r_i \in \mathbb{S}$ as:

$$\mathbf{r}_i = \{\mathbf{f}_i, \boldsymbol{\sigma}_i, \mathbf{b}_i, \mathbf{D}_i^{(c)}, \mathbf{D}_i^{(p)}, D_i^{(t)}\} \tag{1}$$

Fig. 1 shows an overview of the physical infrastructure alongside two SFC requests and their placements.

B. RESOURCE FRAGMENTATION

Fig. 1 additionally depicts instances of resource fragmentation within the network. Without loss of generalization, let us assume that the placement of the two SFC requests saturates the hosting servers and physical links. Under this assumption, servers n_2 and n_3 contribute to compute resource fragmentation since they are available but cannot be used due to bandwidth saturation of the adjacent links. Such isolation of resources reflects a misalignment between a server's potential and its role in the network's active traffic flow, leading to suboptimal resource exploitation. In contrast, physical link (o_2, o_3) showcases communication link fragmentation as, despite its bandwidth availability, it is inaccessible because the surrounding servers and links have reached their resource limits. This condition creates bottlenecks that prevent efficient data transit, even when the network has latent capacity.

To quantify the degree of fragmentation in the infrastructure, we adopt the Resource Fragmentation Degree (RFD) metric first introduced in [8]. This metric is designed to quantitatively measure the status of resource fragmentation at substrate nodes and links by assessing how scattered or isolated resources are in relation to their neighboring entities, either nodes or links. The concept builds on principles from graph theory, where the notion of connectivity describes the robustness of a graph. The principle behind this metric is that the resource availability of a server (or link) is determined by the remaining link and server resources around it.

To calculate the RFD, two additional metrics need to be determined: i) the connectivity of substrate nodes, κ_n, and ii) the connectivity of substrate links, $\kappa_{(u,v)}$. The former measures how "connected" a server $n \in \mathbb{N}$ is as follows:

$$\kappa_n = \frac{1}{\delta_n^{\tau}} \sum_{n' \neq n}^{\mathbb{N}} \rho'_n \, \eta_{n',n}^{\tau}, \tag{2}$$

where, δ_n^{τ} represents the number of nodes whose distance from server n is no longer than τ hops, ρ'_n is the residual ratio of compute resources capacity of server n' (ratio between available and total resources), and $\eta_{n',n}^{\tau}$ is the residual bandwidth ratio of the path between n' and n that is no longer than τ hops (ratio between available and total bandwidth). Similarly, $\kappa_{(u,v)}$ measures the "connectivity" of a link $(u, v) \in \mathbb{L}$. To avoid overloading the notation, we reuse symbols from Eq. (2) for analogous metrics, and we define $\kappa_{(u,v)}$ as:

$$\kappa_{(u,v)} = \frac{1}{\delta_{(u,v)}} \sum_{(u',v') \neq (u,v)}^{\mathbb{L}} \rho_{(u',v')} \, \eta_{(u',v'),(u,v)}, \tag{3}$$

where $\delta_{(u,v)}$ is the number of adjacent links to (u, v), i.e., the links that share a node with (u, v), $\rho_{(u',v')}$ is the residual bandwidth ratio of the adjacent link (u', v'), and $\eta_{(u',v'),(u,v)}$ is the residual ratio of compute resources capacity on the shared node between link (u', v') and link (u, v). The RFD for both servers and physical links is then computed as the complement of their respective connectivities, as follows:

$$\zeta_n = 1 - \kappa_n, \tag{4}$$

$$\epsilon_{(u,v)} = 1 - \kappa_{(u,v)}. \tag{5}$$

Higher values indicate higher degrees of fragmentation.

C. PROBLEM FORMULATION

The problem we solve in this work is described as follows: considering the physical network of the infrastructure and an estimation of the incoming number of SFC requests, their type, and their expected entry point/gateway, find the VNF placement that minimizes network resource fragmentation alongside with resource utilization. This solution should satisfy the constraints of both the infrastructure provider and the users requesting the SFCs. In this direction, we also make the following assumption that leads our formulation and approach to solving VNF-CPP: when considering where

to place VNFs, both the network's edge and cloud compartments provide distinct advantages. On the one hand, the centralized and resilient cloud allows for better resource aggregation, dynamic allocation, and better adaptation to changing traffic patterns and service demands. On the other hand, the edge significantly reduces service latency and promotes localized data processing for quick decision-making and effective traffic offloading, potentially reducing cloud congestion. Though necessary for providing low latency and real-time processing, we consider edge resources scarcer and more "expensive" than those in the centralized cloud. Moreover, we assume that the network's edge can be easily clogged because of the bottleneck that can form when many services try to use edge resources simultaneously. In combination with the increased hardware heterogeneity and diversity at the edge, placing an SFC there poses an increased risk of fragmenting the network's resources.

We formulate the above problem as an Integer Linear Programming (ILP) problem. To this end, two binary decision variables are introduced for the placement of an SFC $\mathbf{r}_i \in \mathbb{S}$; i) $x_{i,k,n}$ that is set equal to 1 if the VNF $f_{i,k} \in \mathbf{f}_i$ is placed on server $n \in \mathbb{N}$ and ii) $y_{(u,v)}^{i,k,k'}$ which is set equal to 1 if the virtual link between $f_{i,k}$ and $f_{i,k'} \in \mathbf{f}_i$ is routed over the physical link $(u, v) \in \mathbb{L}$. Hence, the fragmentation-sensitive objective function for the VNF-CPP can be expressed as:

$$O = \sum_{i=1}^{|\mathbb{S}|} \Big(\sum_{k}^{|\mathbf{f}_i|} \sum_{n}^{\mathbb{N}} \mu_n \, \zeta_n \, x_{i,k,n} + \sum_{k}^{|\mathbf{f}_i|} \sum_{k'}^{|\mathbf{f}_i|} \sum_{(u,v)}^{\mathbb{L}} \epsilon_{(u,v)} \, y_{(u,v)}^{i,k,k'} \Big), \quad (6)$$

where μ_n is a binary cost for server $n \in \mathbb{N}$. In particular, μ_n takes the value of 1 if n is an edge server and 0 otherwise, reflecting in this way the higher cost and increased risk of fragmentation of the edge resources. We note that, despite reducing the emphasis on cloud resource fragmentation optimization, we do include cloud resources in the VNF-CPP optimization for completeness; this way we ensure that SFCs can leverage the cloud's abundant resources as a contingency option, thereby complementing the edge in a realistic setting. Furthermore, as part of our first contribution, incorporating the RFD metrics into the objective function guarantees that the solution not only minimizes resource consumption and communication costs but also optimally distributes the VNFs, resulting in reduced resource fragmentation. Thus, hereafter we will refer to the evaluation of this objective function as the *fragmentation cost*. The placement of a batch of requested SFCs $\mathbb{S}$ is denoted as:

$$\boldsymbol{\beta} = \{ x_{i,k,n}, y_{(u,v)}^{i,k,k'} \; || \; \forall i [0, |\mathbb{S}|], \\ \forall k, k' \in [0, |\mathbf{f}_i|], \forall n \in \mathbb{N}, \forall (u, v) \in \mathbb{L} \}, \quad (7)$$

and the family of all possible placements is denoted as $\mathbb{B}$. We also identify the following constraints in our system:

$$\sum_{k}^{|\mathbf{f}_i|} \Big(\sum_{n}^{\mathbb{N}} d_{i,k,n}^{(c)} x_{i,k,n} + \sum_{k'}^{|\mathbf{f}_i|} \sum_{(u,v)}^{\mathbb{L}} d_{(u,v)}^{(p)} y_{(u,v)}^{i,k,k'} \Big) \leq D_i^{(t)}, \\ \forall i \in [0, |\mathbb{S}|] \quad (8)$$

$$\sum_{n}^{\mathbb{N}} d_{i,k,n}^{(c)} \, x_{i,k,n} \leq D_{i,k}^{(c)}, \forall i \in [0, |\mathbb{S}|], \\ \forall k \in [0, |\mathbf{f}_i|], \quad (9)$$

$$\sum_{(u,v)}^{\mathbb{L}} d_{(u,v)}^{(p)} y_{(u,v)}^{i,k,k'} \leq D_{i,k,k'}^{(p)}, \forall i \in [0, |\mathbb{S}|], \\ \forall k \neq k' \in [0, |\mathbf{f}_i|]. \quad (10)$$

Constraint (8) guarantees that the end-to-end delay demand is respected for every placed SFC. Constraints (9) and (10) make sure that the individual processing delay demands for each VNF and the individual propagation delay demand for each virtual link of an SFC are respected. Also, the computation and communication resources of the infrastructure must not be oversubscribed (constraints (11) and (12) respectively):

$$\sum_{i=1}^{|\mathbb{S}|} \sum_{k}^{|\mathbf{f}_i|} \boldsymbol{\sigma}_{i,k} \, x_{i,k,n} \leq \mathbf{R}_n, \forall n \in \mathbb{N}, \quad (11)$$

$$\sum_{i=1}^{|\mathbb{S}|} \sum_{k}^{|\mathbf{f}_i|} \sum_{k'}^{|\mathbf{f}_i|} b_{i,k,k'} (y_{(u,v)}^{i,k,k'} + y_{(v,u)}^{i,k,k'}) \leq B_{(u,v)}, \; \forall (u, v) \in \mathbb{L}. \quad (12)$$

The flow conservation constraint (13), ensures the interconnection of the VNFs by performing the routing between the source and destination of each virtual link.

$$\sum_{v \in \mathbb{H}} (y_{(u,v)}^{i,k,k'} - y_{(v,u)}^{i,k,k'}) \\ = x_{i,k,u} - x_{i,k',u}, \; \forall u \in \mathbb{H}, \forall i \in [0, |\mathbb{S}|], \forall k \neq k' \in [0, |\mathbf{f}_i|]. \quad (13)$$

Finally, we need to ensure that each VNF will be placed at only one node, with constraint (14), provided that the specific node is a server, constraint (15). We should also ensure that the first VNF of an SFC is placed on the entry gateway of the user requesting the SFC, as designated in constraint (16).

$$\sum_{h \in \mathbb{H}} x_{i,k,h} = 1, \; \forall i \in [0, |\mathbb{S}|], \; \forall k \in [0, |\mathbf{f}_i|], \quad (14)$$

$$\sum_{n \in \mathbb{N}} x_{i,k,n} = 1, \; \forall i \in [0, |\mathbb{S}|] \; \forall k \in [1, |\mathbf{f}_i|], \quad (15)$$

$$\sum_{q \in \mathbb{W}_{\mathbb{Q}}} x_{i,0,q} = 1, \; \forall i \in [0, |\mathbb{S}|]. \quad (16)$$

To this end, the offline optimization problem for minimizing resource fragmentation while successfully placing a batch of incoming SFC requests can be formulated as:

$$P(1): \min_{\boldsymbol{\beta} \in \mathbb{B}} \quad (6) \quad (17a)$$

$$\text{s.t. } (8) - (16) \quad (17b)$$

As an extension of the classic VNF-CPP, this fragmentation-aware formulation falls into the category of the virtual network embedding problems, thus it is classified as NP-hard [3]. This means that traditional ILP-based methods fall short of providing real-time solutions, especially in highly

TABLE 2. Summary of the key notation.

Notation	Description
$\mathbb{G}=(\mathbb{H},\mathbb{L})$	Physical infrastructure graph (nodes, links)
$\mathbb{N}=\mathbb{N}_{\mathbb{E}}\cup\mathbb{N}_{\mathbb{C}}$	Servers set consisting of edge and cloud subsets
$\mathbb{W}=\mathbb{W}_{\mathbb{Q}}\cup\mathbb{W}_{\mathbb{O}}$	Routers set consisting of gateway and traffic subsets
$\mathbf{R}_n$	Server's available resources (CPU, memory, storage)
$B_{(u,v)}$	Link's bandwidth capacity
$d^{(p)}_{(u,v)}$	Link's propagation delay
$\mathbf{r}_i \in \mathbb{S}$	SFC request
$\mathbf{f}_i=\{f_{k,\lvert\mathbf{f}_i\rvert}\}$	VNF set of SFC request
$\boldsymbol{\sigma}_i$	SFC request's resource demands per VNF
$\mathbf{D}^{(c)}_i$	SFC request's max processing delays per VNF
$d^{(c)}_{i,k,n}$	VNF's incurred processing delay on server
$\mathbf{b}_i$	SFC request's bandwidth demands per virtual link
$\mathbf{D}^{(p)}_i$	SFC request's max propagation delays per virtual link
$D^{(t)}_i$	SFC request's max end-to-end delay
$\zeta_n,\ \epsilon_{(u,v)}$	Server and physical link RFD
$x_{i,k,n}$	Binary variable set to 1 if VNF is placed on server
$y^{i,k,k'}_{(u,v)}$	Binary variable set to 1 if virtual link is hosted over physical link
μ_n	Binary cost set to 1 for edge servers
$\boldsymbol{\beta} \in \mathbb{B}$	SFC requests placement
$\mathbf{s}, \mathbf{a}, \mathcal{C}, \mathcal{R}$	CMDP state, action, cost and reward
c_ϕ	Degree of constraint violation
$\pi \in \Pi$	SFC request placement policy
$J^{\pi}_{\mathcal{R}}$	Expected cumulative reward under placement policy
$J^{\pi}_{c_\phi}$	Expected degree of constraint violation under placement policy
L^{θ}_{λ}	Lagrangian
$\lambda_\phi \in \boldsymbol{\lambda}$	Lagrange multipliers
$\theta,\ \theta'$	NCO-RCPO agent's weights
$\psi_w,\ \psi_1,\ \psi_2$	θ', θ and λ learning rates

dynamic environments as the one envisioned. We note that the key notation used for the system model and subsequent mathematical formulation is provided in Table 2.

IV. ONLINE FRAGMENTATION-AWARE VNF PLACEMENT USING NCO AND RCPO

To efficiently solve problem (17), we seamlessly combine the powers of Neural Combinatorial Optimization (NCO) [10] and Reward Constrained Policy Optimization (RCPO) [9] into an online, fragmentation-aware VNF placement mechanism for edge/cloud infrastructures. In this context, the Constrained Markov Decision Process (CMDP) serves as a suitable modeling framework. This formulation allows us to incorporate constraints into the decision-making process, ensuring that while the NCO agent learns to maximize its rewards, it also respects the system's constraints.

A. CONSTRAINT MARKOV DECISION PROCESS FORMULATION

In classic Markov Decision Processes (MDP), an agent interacts with a stochastic environment aiming to optimize a cumulative reward over time. The agent is characterized by the tuple $\{\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma\}$, where $\mathcal{S}$ is the set of states in the environment, $\mathcal{A}$ is the set of possible actions for the agent, $\mathcal{P}$ is the next state's transition probability function, $\mathcal{R}$ is a function which represents the immediate reward received by the agent and $0 < \gamma < 1$ is the discount factor that determines the present effect of future rewards. A CMDP extends the MDP framework by introducing additional constraints on the expected cumulative cost of certain actions or state transitions. Here we have the augmented tuple $\{\mathcal{S}, \mathcal{A}, \mathcal{C}, \alpha, \mathcal{P}, \mathcal{R}, \gamma\}$, where the cost function $\mathcal{C}$ quantifies the expense or penalty of taking an action while transitioning to the next state. The threshold value α denotes the system tolerance before penalties are applied. We consider the decision-making slotted in timeslots $t \in \mathbb{Z}_+$. As the state in our case is high-dimensional, the transition probabilities $\mathcal{P}$ cannot be accurately obtained. Thus, the CMDP is simplified into a model-free process $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{C}, \alpha, \mathcal{R}, \gamma\}$ with the following components:

1) STATE

At each decision-making time t, the agent observes the system state. Before defining the system state, we need to introduce two auxiliary variables, $\mathbf{R}'$ and $\mathbf{B}'$ which contain the currently available resources for every server $n \in \mathbb{N}$ and available bandwidth capacity for every physical link $(u, v) \in \mathbb{L}$. Using Eqs. (11) and (12), we have:

$$\mathbf{R}' = \{\mathbf{R}_n - \sum_{i=1}^{|\mathbb{S}|}\sum_{k}^{|\mathbf{f}_i|} \boldsymbol{\sigma}_{i,k}\, x_{i,k,n} \mid \forall n \in \mathbb{N}\},$$

$$\mathbf{B}' = \{B_{(u,v)} - \sum_{i=1}^{|\mathbb{S}|}\sum_{k}^{|\mathbf{f}_i|}\sum_{k'}^{|\mathbf{f}_i|} b_{i,k,k'}\,(y^{i,k,k'}_{(u,v)} + y^{i,k,k'}_{(v,u)}) \mid \forall (u,v) \in \mathbb{L}\}.$$

We design our online placement mechanism to place *only one requested SFC at each timeslot*. Therefore, we drop the notation $i \in |\mathbb{S}|$. We now define the system state $\boldsymbol{s}_t \in \mathcal{S}$ as a combination of information regarding the currently requested SFC, the previously placed SFCs, and the state of the physical network in terms of resource availability:

$$\mathbf{s}_t = \{\mathbf{r}, \boldsymbol{\beta}, \mathbf{R}', \mathbf{B}'\}, \tag{18}$$

2) ACTION

The action $\mathbf{a}_t \in \mathcal{A}$ taken by the agent at timeslot t concerns the placement of the VNFs $\mathbf{f}$ of SFC request $\mathbf{r}$:

$$\mathbf{a}_t = \{x_{k,n},\ y^{k,k'}_{(u,v)} \mid \forall k, k' \in [0, |\mathbf{f}|], \forall n \in \mathbb{N}, \forall (u,v) \in \mathbb{L}\}, \tag{19}$$

3) COST

The cost in our environment reflects the *degree of constraint violations*. Assuming a system constraint of the generic form $\phi : Ax \leq B$ then the degree of constraint violation is given as

$c_\phi = B - Ax$. Therefore, we formulate the cost of our system as the aggregated degree of constraint violations that occur due to performing action $\mathbf{a}_t$ while in state $\mathbf{s}_t$ and transitioning to state $\mathbf{s}_{t+1}$:

$$\mathcal{C}(\mathbf{s}_t, \mathbf{a}_t) = \sum_{\phi=\,(8)}^{(16)} max\{0, c_\phi - \alpha \mid \mathbf{s}_t, \mathbf{a}_t\}. \tag{20}$$

Since we want the placement policy to result in actions that do not violate any constraints, we set $\alpha = 0$. An action that results in a cost greater than zero is considered *infeasible*.

4) REWARD

After performing action $\mathbf{a}_t$ on state $\mathbf{s}_t$, the agent gets a feedback which directly determines its strategy. As our optimization objective is to minimize the fragmentation in the network infrastructure while respecting all the identified constraints, we formulate the reward as the scaled inverse of the sum of the fragmentation cost obtained from Eq. (6):

$$\mathcal{R}(\mathbf{s}_t, \mathbf{a}_t) = Z\, exp(O \mid \mathbf{s}_t, \mathbf{a}_t)^{-1}, \tag{21}$$

where $Z \in \mathbb{R}$ is an empirically selected tuning coefficient that adjusts the magnitude of the reward to ensure it is appropriately balanced for the agent's training. This reward form emphasizes the impact of fragmentation costs; small decreases in fragmentation cost lead to significant increases in the reward, encouraging actions that reduce fragmentation even further.

B. LAGRANGE RELAXATION & NCO-RCPO-BASED SOLUTION

The first step to solving the CMDP formulation is to define the primal problem, the solution of which will pinpoint the *optimal placement policy* $\pi(\mathbf{a} \mid \mathbf{s}) \in \Pi$ that maximizes the expected cumulative reward over time, $J^\pi_\mathcal{R} = \mathbb{E}_\pi\,[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(\mathbf{s}_t, \mathbf{a}_t)]$, while ensuring that the expected cumulative cost $J^\pi_\mathcal{C} = \mathbb{E}_\pi\,[\sum_{t=0}^{\infty} \gamma^t \mathcal{C}(\mathbf{s}_t, \mathbf{a}_t)]$ does not exceed the system's tolerance. The latter translates directly in the requirement that the expected degrees of constraint violations, $J^\pi_{c_\phi} = \mathbb{E}_\pi\,[\sum_{t=0}^{\infty} \gamma^t c_\phi]$ do not exceed the threshold $\alpha = 0$:

$$P(2): \max_{\pi \in \Pi} \quad J^\pi_\mathcal{R} \tag{22a}$$

$$\text{s.t.} \quad J^\pi_{c_\phi} \leq 0,\ \forall \phi \in (8) - (16). \tag{22b}$$

To solve the above primal problem, we implement an NCO agent by using a Sequence-to-Sequence (*S2S*) neural network consisting of a Long Short-Term Memory (LSTM) encoder and a decoder. In order to significantly reduce the action space dimensions and accommodate the applicability of NCO to our environment, we decompose the action calculation of Eq. (19) into two stages; in the first stage, the agent produces $\mathbf{a}_t$, i.e., a sequence of servers $n \in \mathbb{N}$ where the VNFs of the requested SFC will be placed into. In the second stage, the well-known Dijkstra's Algorithm produces the routing decisions by calculating the shortest paths, i.e., the physical links that will be utilized to interconnect the SFC's VNFs $f \in \mathbf{f}$. Using the S2S model allows for using the Bahdanau attention mechanism that improves the agent's efficiency as we are now mapping the input sequences of the request's VNFs to the output sequences of the infrastructure's servers [27]. The neural network denoted by its weights θ infers the emerging parameterized placement policy $\pi_\theta(\mathbf{a} \mid \mathbf{s})$.

For the NCO agent to work, we additionally need to integrate the expected constraint violation costs of the primal problem into the objective function. To do so, we utilize the Lagrange relaxation technique. The primal problem is thus transformed into the following dual unconstrained optimization problem:

$$P(3): \min_{\forall \lambda_\phi \in \lambda} \max_{\theta} L^\theta_\lambda = \min_{\forall \lambda_\phi \in \lambda} \max_{\theta} [J^{\pi_\theta}_\mathcal{R} - \sum_{\phi=\,(8)}^{(16)} \lambda_\phi J^{\pi_\theta}_{c_\phi}], \tag{23}$$

where L^θ_λ is the Lagrangian, representing the penalized reward, and $\lambda_\phi \geq 0, \lambda_\phi \in \lambda$ are the Lagrange multipliers (penalty coefficients) for the constraints c_ϕ. We notice that as these multipliers grow, the solution approaches this of the primal problem, i.e., Eq. (22). Here, we identify a weakness in related works in the literature, for example, in [17], where the Lagrange multipliers are selected manually. To improve the efficiency of this procedure, we incorporate the RCPO technique in our mechanism, a multi-timescale approach for constrained policy optimization that uses a penalty signal to guide the policy towards a constraint-satisfying one and eventually selects the Lagrange multipliers automatically. As the guiding penalty signal we choose the expected weighted constraint violation degree $\sum_{\phi=\,(8)}^{(16)} \lambda_\phi\, J^{\pi_\theta}_{c_\phi}$, the second term of the Lagrangian L^θ_λ.

Calculating the weights θ together with the multipliers λ_ϕ requires training our agent once and offline, using a training dataset of $\mathbb{S}'$ SFC requests in batches of $\mathbb{B} \subset \mathbb{S}'$. According to the RCPO suggestions, we introduce a two-timescale training process that involves fast and slow update intervals: the weights θ are updated through the guidance of Eq. (23) on the fast timescale, whereas on the slow timescale the λ_ϕ multipliers are increased gradually until the constraint violation degree vanishes. When iteratively repeated this procedure results in convergence as proven in [9]. We use $\dot{t}$ to denote an epoch of the training process.

On the fast timescale, in order to compute the S2S weights θ of the NCO agent that optimizes L^θ_λ, we use the Monte-Carlo Policy Gradients and the gradient descent as follows:

$$\theta_{\dot{t}+1} = \theta_{\dot{t}} + \psi_1(\dot{t})\, \nabla_\theta L^{\theta_{\dot{t}}}_{\lambda_{\dot{t}}}, \tag{24}$$

where $\psi_1(\dot{t}) \in [0, 1]$ is the fast timescale learning rate. To calculate the gradient of the Lagrangian, we use the log-likelihood approach [28]:

$$\nabla_\theta L^\theta_\lambda = \mathbb{E}_{\pi_\theta}[\,[\,\mathcal{R} - \sum_{\phi=\,(8)}^{(16)} \lambda_\phi\, c_\phi\,]\, \nabla_\theta\, log\, \pi_\theta \mid \mathbf{s}, \mathbf{a}\,]. \tag{25}$$

Since the exact computation of the gradient is a computationally challenging task in a highly dimensional environment like ours, we approximate it with Monte-Carlo sampling:

$$\nabla_\theta L_\lambda^\theta \approx \frac{1}{|\mathbb{B}|}\sum_{\mathbf{r}}^{\mathbb{B}}[(\mathcal{R} - \sum_{\phi=\,(8)}^{(16)} \lambda_\phi c_\phi - w_{\theta'})\nabla_\theta log\ \pi_\theta\ |\mathbf{s}:\mathbf{r}, \mathbf{a}:\mathbf{r}], \tag{26}$$

where $w_{\theta'}$ is the *baseline estimator* [29]; the baseline is a state-dependent supporting model that estimates the reward minus the cost yielded as a result of the placement action of the NCO-RCPO agent under the current placement policy. This estimator helps reduce variance, accelerate convergence, and improve sample efficiency during training. We implement it as an auxiliary LSTM encoder connected to a feedforward Multilayer Perceptron (MLP) output layer with weights θ'. The baseline is trained using gradient descent to minimize the Mean Squared Error (MSE) between the predicted and the real values:

$$\text{MSE}_w^{\theta'} = \frac{1}{|\mathbb{B}|}\sum_{\mathbf{r}}^{\mathbb{B}} ||(\mathcal{R} - \sum_{\phi=\,(8)}^{(16)} \lambda_\phi\, c_\phi) - w_{\theta'} \mid \mathbf{s}:\mathbf{r}, \mathbf{a}:\mathbf{r}||^2. \tag{27}$$

We update the baseline's weights on a timescale faster than the one for the S2S weights as follows:

$$\theta'_{i+1} = \theta'_i - \psi_w(i)\, \nabla_{\theta'} \text{MSE}_w^{\theta'}, \tag{28}$$

where $\psi_w(i) \in [0, 1]$ is the baseline's learning rate and $\nabla_{\theta'}\text{MSE}_w^{\theta'}$ is the gradient of Eq. (27) with respect to the weights θ'. On the slow scale, we update the Lagrange multipliers as follows:

$$\lambda_\phi^{i+1} = \Gamma[\lambda_\phi^i - \psi_2(i)\, \nabla_{\lambda_\phi} L_{\lambda_i}^{\theta_i}], \forall \phi \in (8) - (16), \tag{29}$$

$$\nabla_{\lambda_\phi} L_\lambda^\theta = -J_{c_\phi}^{\pi_\theta}, \tag{30}$$

where $\psi_2(i) \in [0, 1]$ is the slow timescale learning rate and Γ restricts the value of λ_ϕ by projecting it into the range $[0, \Lambda]$, $\Lambda \gg 0$. This projection prevents extreme updates that could destabilize the optimization process.

C. NCO-RCPO AGENT TRAINING

Algorithm 1 formally describes the steps of the training process for the fragmentation-aware NCO-RCPO agent. The goal is to train the agent to optimally place the VNFs of an SFC request in the network, considering both the fragmentation cost and the adherence to the system's constraints. The input is a training dataset $\mathbb{S}'$ of SFC requests and the size of each training batch $|\mathbb{B}|$ (mini-batch training). At first, the weights of the two S2S neural networks are randomized, and the values of the Lagrange multipliers are set to zero. During the training, at each epoch i the gradients are initially set equal to zero to avoid summing them over multiple batches. Then, a batch of SFC requests $\mathbb{B}$ is sampled, and for each request $\mathbf{r}$, the system state is computed, a placement action is performed, the reward is gathered, and the constraint violation penalties are calculated. Having this information

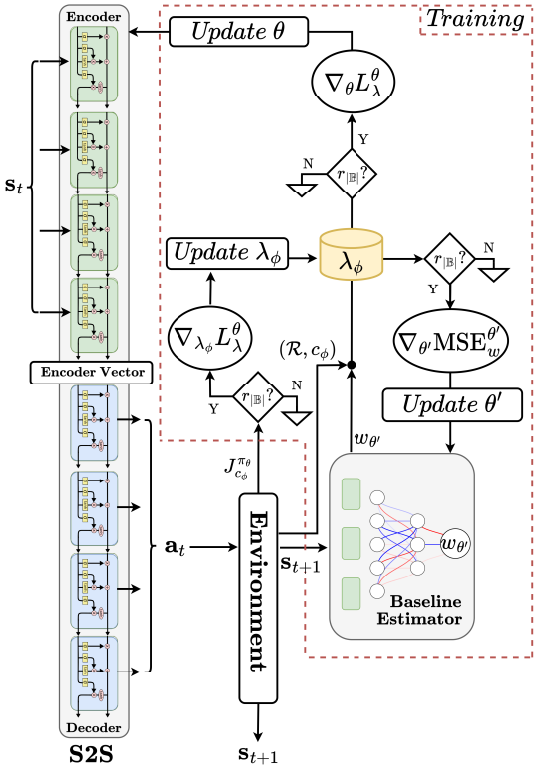


FIGURE 2. NCO-RCPO agent overview.

available, the weight and Lagrange multiplier-related gradients are computed; then, the weights of the agent's neural networks and the Lagrange multipliers are updated accordingly. The algorithm returns the agent's trained weights. Fig. 2 gives an overview of the NCO-RCPO agent and the training process.

D. GREEDY CORRECTION HEURISTIC – RUNTIME ROBUSTNESS

As with every RL agent, the expected penalty when the NCO-RCPO agent has converged is zero with slight deviations [30]. Therefore, theoretically, during the online phase, the agent could make infeasible placement decisions. To address these potential infeasibilities and robustify the operation of our framework, we propose the *Greedy Correction* (GC) heuristic that complements the agent in runtime with an error correction mechanism. Here, the infeasible placement is used as a starting point, and we incrementally improve it in a step-by-step, best-effort manner until it becomes feasible. First, the servers are sorted in ascending order based on their available resources. Then, for every placed VNF in the infeasible action, we change the selected server to one of the sorted ones sequentially, starting from the one with the least available resources. This selection criterion serves as a computationally cheap way of keeping the fragmentation cost low during this infeasibility repair. If the new placement reduces the constraint violation cost, we update the action accordingly. We repeat for every placed

Algorithm 1 NCO-RCPO Agent Training

Input: SFC Training dataset $\mathbb{S}'$, batch size $|\mathbb{B}|$
Initialize: randomize θ, θ' and $\lambda_\phi = 0, \forall \phi \in$ (8) − (16)
for $i = 0, 1, 2, \ldots$ **do**
  Reset gradients $\nabla_\theta L_\lambda^\theta$, $\nabla_{\theta'} \text{MSE}_w^{\theta'}$, $\nabla_{\lambda_\phi} L_\lambda^\theta$;
  Sample batch $\mathbb{B}$;
  for $\mathbf{r} \in \mathbb{B}$ **do**
    Compute the system state $\mathbf{s}$;
    Perform action $\mathbf{a}$ based on policy π_θ;
    Compute reward $\mathcal{R}$;
    Compute constrained violation degrees $c_\phi, \forall \phi$;
  Compute $\nabla_{\theta'} \text{MSE}_w^{\theta'}$ using Eq. (27);
  Compute $\nabla_\theta L_\lambda^\theta$ using Eq.(26);
  Compute $\nabla_{\lambda_\phi} L_\lambda^\theta, \forall \phi$ using Eq. (30);
  Update θ', θ, λ_ϕ.
Return θ

Algorithm 2 FANCORP Online Operation

Input: NCO-RCPO agent's trained weights θ
for $t = 0, 1, 2, \ldots$ **do**
  Receive SFC request $\mathbf{r}_i^t$;
  Compute the system state $\mathbf{s}_t$;
  Select action $\mathbf{a}_t$ based on policy π_θ;
  if $\mathcal{C}(\mathbf{s}_t, \mathbf{a}_t) == 0$ **then** `//Feasible Placement`
    Perform action $\mathbf{a}_t$;
  else `// Greedy Correction`
    while $(k \leq |\mathbf{f}_i|)$ && $(\mathcal{C}(\mathbf{s}_t, \mathbf{a}_t) > 0)$ **do**
      $\mathbb{N}' \leftarrow \mathbb{N}$ sorted in ascending $\mathbf{R}'_n$ order;
      repeat
        Pop server $n' \in \mathbb{N}'$;
        Place VNF $f_{i,k}$ on $n' \Rightarrow \mathbf{a}'_t, \mathbf{s}'_t$;
      until $\mathcal{C}(\mathbf{s}_t, \mathbf{a}'_t) < \mathcal{C}(\mathbf{s}_t, \mathbf{a}_t)$;
      $\mathbf{a}_t \leftarrow \mathbf{a}'_t$, $k \leftarrow k + 1$;
    if $\mathcal{C}(\mathbf{s}_t, \mathbf{a}_t) == 0$ **then**
      Perform action $\mathbf{a}_t$;
    else
      Reject SFC request $\mathbf{r}_i^t$;

VNF or until the placement action becomes feasible. If the resulting action is still infeasible, the request is rejected. This fast and simple heuristic solidifies the proposed framework with a backup plan. In this way, we provide an NCO-RCPO agent that ensures robust, high-quality, fragmentation-aware SFC placement, which importantly can also mitigate its own miscalculations; this concludes the second contribution of this work.

E. FANCORP FRAMEWORK OVERVIEW

In this subsection, we summarize the online operation of the proposed FANCORP framework. Once an SFC request arrives, the system state is observed, and the NCO-RCPO agent takes a server placement decision for each one of the requested VNFs, using its fragmentation-aware learned policy. Then, the well-known Dijkstra's Algorithm is used to find the shortest paths that interconnect the SFC's VNFs, completing in this way the placement action. At this point, the feasibility of the action is checked; if the placement violates any constraints, the GC heuristic is triggered and attempts to repair the infeasibility. Should it fail to do so, the request is rejected. Algorithm 2 gives a detailed outline of the proposed framework's operation.

V. NUMERICAL RESULTS

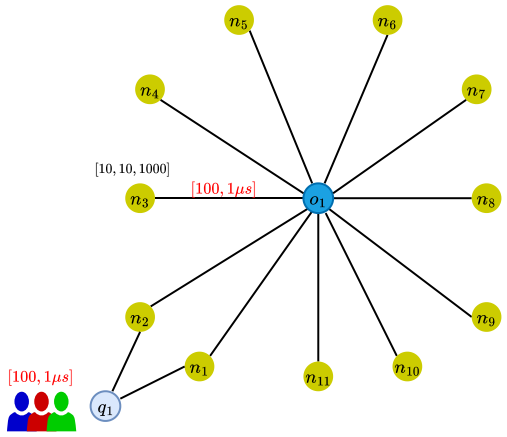
In this section, we evaluate the efficiency of the proposed FANCORP framework through extensive simulation. First, we present the experimentation setup. Then, we compare the proposed NCO-RCPO-based Lagrange multiplier learning technique against a related work. Finally, the whole operation of the FANCORP framework is evaluated and benchmarked against state-of-the-art solutions from the literature.

A. EXPERIMENTATION SETUP

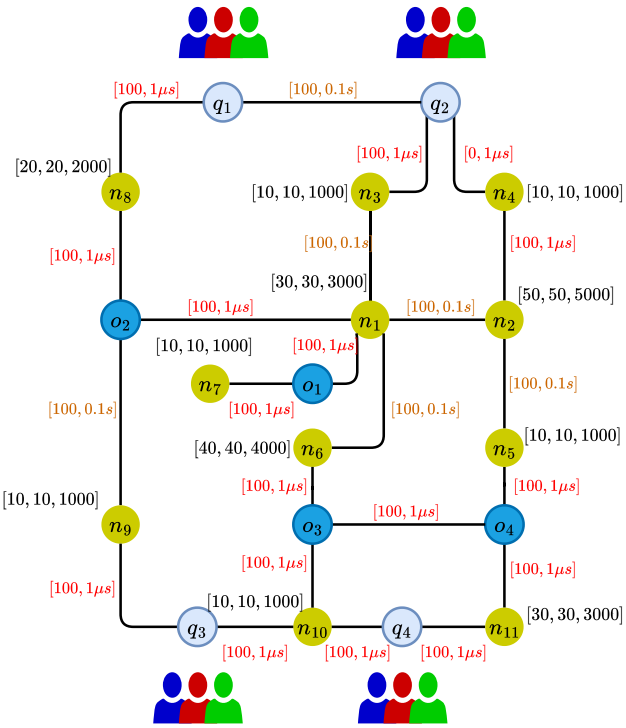
We perform our simulations on four representative network topologies: a simple *star network* that was used in [17] (for one-to-one comparison purposes), depicted in Fig. 3a; a more realistic *metropolitan network* topology, depicted in Fig. 3b which was provided by an industrial partner and represents a typical edge/cloud infrastructure [31]; and two additional topologies, *Abilene* and *New York*, sourced from the SNDlib library [32].

In detail, the Star network consists of $|\mathbb{N}| = 11$ homogeneous edge servers equipped with 10 CPU cores, 10 *GB* of RAM and 1000 *GB* of storage, and $|\mathbb{W}| = 2$ routers, one of which serves as a gateway and one as a traffic router. Each physical link has a bandwidth capacity of 100 *Mbps* and induces a propagation delay of 1 μs. On the other hand, the Metropolitan network is designed to reflect the complexity and variability of urban network infrastructures. In our adaptation, the topology has been slightly modified from the real one to simplify connectivity between sites. However, the network configuration and resource distribution still reflect a real operational environment, providing a solid basis for evaluating VNF placement strategies on realistic scenarios that our industrial partner has encountered. It consists of $|\mathbb{N}|$=11 heterogeneous servers (n_2 and n_6 are cloud servers) the resources of which range between $10-50$ CPU cores, $10-50$ *GB* of RAM and $1000-5000$ *GB* of storage capacity. Additionally, $|\mathbb{W}| = 8$ routers are found in this network, 4 of which serve as gateways and 4 as traffic routers. Each physical link has a bandwidth capacity of 100 *Mbps* and induces a propagation delay that ranges between 1 $\mu s - 0.1$ *s*. For further details, readers can refer to the figures.

The Abilene network is a backbone network with high-capacity links and a few highly connected nodes for long-distance communication across the U.S. In contrast, the New York network is an urban network with a dense topology,



(a) Star Network.



(b) Metropolitan Network.

FIGURE 3. Network topologies used for the evaluation.

more nodes, and high traffic capacity within a metropolitan area. Including the Abilene network aids in further evaluating our scheme in large-scale backbone infrastructures, while the New York network evaluates the performance in dense urban environments. The Abilene network consists of $|\mathbb{N}| = 12$ servers, $|\mathbb{W}| = 8$ routers, 4 of which are gateway routers, and 15 physical links, while the New York network comprises $|\mathbb{N}| = 16$ servers, $|\mathbb{W}| = 20$ routers, 4 which are gateway routers, and 49 physical links. Similar to the Metropolitan network, each server in both networks is equipped with 10–50 CPU cores, 10–50 *GB* of RAM, and 1000–5000 *GB* of storage capacity. Their physical links have a bandwidth capacity of 100 *Mbps* and a propagation delay ranging between $1\ \mu s - 0.1\ s$.

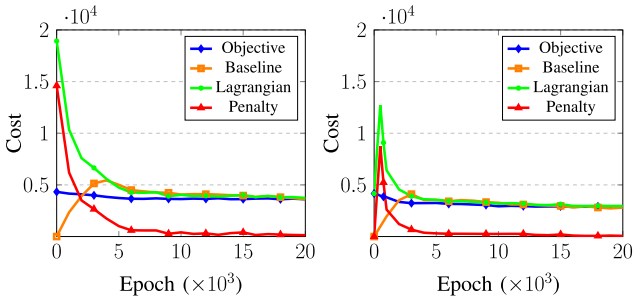
Regarding the SFC requests, we utilize the 3 SFC topologies introduced in [26], with lengths varying from 3 to 5 VNFs, unless stated otherwise. We want our design to emulate Ultra-Reliable Low-Latency Communication (URLLC) scenarios where low latency is crucial. Therefore, each SFC is designed with an end-to-end delay tolerance of 1 *ms*. Each VNF's resource demands range between $1-4$ CPU cores, $1-8\,GB$ of RAM, and $100-200\,MB$ of storage, while its processing delay tolerance ranges between $60-100\ \mu s$. For the virtual links, we consider a propagation delay tolerance of $60-400\ \mu s$ and a bandwidth requirement of $10-20\,Mbps$. The selection of the above parameters is in sync with up-to-date specifications for similar environments [33], [34], [35].

We implement the NCO-RCPO as a four-layer S2S neural network with a hidden size of 128 for each LSTM. The baseline estimator model consists of a four-layer LSTM encoder with a hidden size of 128 as well. Its MLP component has two hidden layers equipped with 128 neurons each. For all the experiments, we train the agent for placing up to $|\mathbb{B}| = 100$ SFC requests in a training session of 30×10^3 epochs. The extensive training of 30×10^3 epochs ensures that the model thoroughly learns the complex patterns for efficient SFC placement across diverse network scenarios. The ψ_1 learning rate is set to 10^{-4} for updating the S2S weights. The baseline model has a significantly higher learning rate of $\psi_w = 10^{-1}$, while the slow learning rate for the Lagrange multipliers is set to $\psi_2 = 10^{-5}$. The Adam optimizer is used to solve Eqs. (24), (28) and (29); specifically, we treat these equations as loss functions and iteratively update the weights in the direction that maximizes Eqs. (24) and (28) and minimizes Eq. (29). To train the agent, we used a Google Cloud Virtual Machine equipped with a system RAM of 83.5 *GB*, a GPU A100 (40 *GB* version) and a storage capacity of 166.8 *GB*. The training lasted 80 *min* for the Star network and 130 *min* for the Metropolitan network. The results were averaged over ten experiment repetitions to reduce variability. For reproducibility, the link to the GitHub repository containing the code implementation used in the simulation is provided.[1]

B. NCO-RCPO-BASED LAGRANGE MULTIPLIERS LEARNING

To demonstrate the efficacy of the NCO-RCPO-based Lagrange multipliers learning, we benchmark it against the methodology proposed by Solozabal et al. in [17], where a similar approach for NCO-based VNF placement was followed, however, with the energy saving criterion as the objective. There, the authors employed the Lagrange relaxation method to transform the primal problem into an unconstrained one, though the selection of Lagrange multipliers was manual. Our aim in this subsection is to showcase the benefits of the automatic, penalty-driven selection; thus, to ensure a consistent basis for comparison, we temporarily adopt energy saving as the objective of our agent and we replicate the dynamic network environment presented in [17].

[1] https://github.com/RamyMo/vnf_placement_using_drl_rcpo



(a) Manual λ Selection [17]. (b) Proposed λ Learning.

FIGURE 4. Benchmarking costs evolution during training.

This environment considers SFCs with varying demands and a length ranging from 12 to 18 VNFs arranged in a linear bus topology and the exact same Star topology network infrastructure. Additionally, instead of maximizing the expected cumulative reward in the primal problem, we alternatively seek to minimize energy consumption, following the approach of the compared work.

Figs. 4a and 4b showcase the VNF placement policy learning process of the NCO agent without and with incorporating RCPO to learn the Lagrange multipliers. Here, the *objective* cost is computed by summing up the energy cost of the VNF placements using the objective function defined in [17]; the *penalty* cost is the sum of all the constraint violation degrees weighted by the Lagrange multipliers; the *Lagrangian* cost is the sum of the energy cost and penalty cost; lastly, the *baseline* is the estimate of the Lagrangian cost, which is obtained using the supporting model $w_{\theta'}$. We observe that in Fig. 4a, initially, all the costs are high, indicating that the system starts with a suboptimal configuration. The agent starts with a high objective cost and penalty, and as it learns, it keeps adjusting its weights to minimize them until they are stabilized.

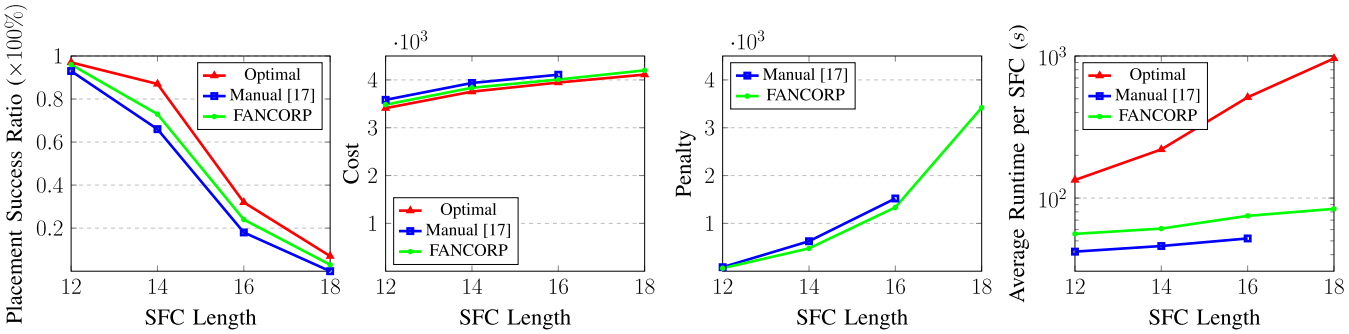
Contrarily, in Fig. 4b, we see that the penalty starts at zero as the initial values of all the Lagrange multipliers are set to zero. Once the Lagrange multipliers are updated for the first time using Eq. (29), the penalty cost sharply increases; however, it decreases quickly over the next few epochs, indicating that the agent learns to avoid constraint violations. Compared to the manual selection, the average costs yielded through the RCPO-based procedure are approximately 20% lower than the manual selection after convergence. Moreover, the costs, particularly the Lagrangian and the penalty, appear to converge more smoothly compared to the behavior observed in Fig. 4a, suggesting a more stable and consistent learning process. This improvement is achieved by involving the penalty signal in the learning process of the NCO agent. We conclude that the manual multipliers selection introduces inefficiencies in the learning process due to reduced adaptability in responding to constraint violations dynamically.

Having trained the two NCO agents (with and without incorporating RCPO), we then assess their VNF placement decisions and additionally compare them against the optimal in terms of SFC placement success ratio, i.e., the ratio of the number of successfully placed SFC requests to the total number of requests made. To derive the optimal placement, we solve the ILP formulation using Gurobi. Furthermore, since benchmarking only concerns the Lagrange multipliers, FANCORP's greedy correction mechanism is deactivated to ensure a fair comparison. Fig. 5a shows the success ratio for SFC requests of increasing average length for the three different placement algorithms. Naturally, this ratio declines for all algorithms as the length grows. This is anticipated as placing longer SFCs becomes increasingly difficult due to the intensified resource and performance demands. Nevertheless, FANCORP outperforms the manual approach across all SFC lengths, scoring an average of 5% higher success ratio.

Moving on, Fig. 5b presents the average objective/energy cost for the different SFC lengths. Again, we observe that as the length grows, the average cost rises for all three algorithms, which is expected since longer SFCs typically necessitate allocating more resources. However, FANCORP consistently exhibits better performance compared to the manual approach across all SFC lengths by yielding an average of 2.5% lower cost while performing close to the optimal (merely 1.5% higher). It is also worth mentioning that the manual approach could not result in a successful placement for any SFCs of length 18, which is why Fig. 5b does not provide cost data for this length. This happens because the longer the SFCs, the higher the likelihood of constraint violations, which highlights the weaknesses of the suboptimal manual selection compared to the proposed automated process.

Fig. 5c illustrates the average constraint violation penalty during the SFC placement when comparing our approach with the manual one. The Gurobi method's results are omitted from this comparison since, as an optimal solver, it either gives a solution for the problem or indicates that a solution is infeasible. The findings here corroborate the proposed method's superiority. We observe that the average penalty increases for both algorithms as the length of the requested SFCs increases for the same reasons as before. However, FANCORP manages to achieve approximately a 18% lower average constraint violation penalty than the manual approach. We note that we do not provide penalty data for SFCs of length 18 for the manual approach due to zero placement success.

Overall, this study demonstrates the positive impact of automatically learning the Lagrange multipliers for VNF-CPP compared to manually selecting them. The latter can be labor-intensive and lacks the adaptability required for dynamic environments, leading to subpar solutions. It is a solution inherently limited by the expertise and intuition of the practitioner and fails to capitalize on the iterative learning potential of DRL methodologies, which can discover complex patterns that lead to improved multiplier values. The proposed automated learning process enables the framework to dynamically adapt to changing network conditions and constraints based on real-time feedback from the environment, ensuring a more robust and flexible approach to VNF



(a) Placement Success Ratio. (b) Objective Cost. (c) Constraints Violation. (d) Average Runtime per SFC.

FIGURE 5. FANCORP vs. Solozabal et al. (Manual) [17] vs. optimal.

placement. As a last remark, while the ILP-based approach, Gurobi, serves as a valuable baseline for comparison, its applicability in large-scale, dynamic VNF placement scenarios is limited. This is due to the inherent computational complexity of exhaustively solving ILP problems, as hinted in Fig. 5d, where the average runtime per SFC placement is depicted. Once again we note that the plot does not include data for the manual approach at an SFC length of 18 because the placement success ratio for the manual plot is zero. All in all, FANCORP offers a real-time, scalable, and near-optimal solution.

C. RESOURCE FRAGMENTATION MINIMIZATION

In the following, we evaluate the efficiency of the FANCORP framework in solving the fragmentation-aware VNF-CPP. For this purpose, we perform two families of experiments, one on the Star network and one on the Metropolitan network. First, we examine the training and how the different parameters, as well as the infrastructure complexity, affect the fragmentation cost achieved. Then, we provide insights into the impact the greedy correction mechanism has on the overall robustness of FANCORP. The parameter τ that determines the maximum hops for measuring node and link connectivity when calculating the network's RFD is set to 2.

1) STAR NETWORK

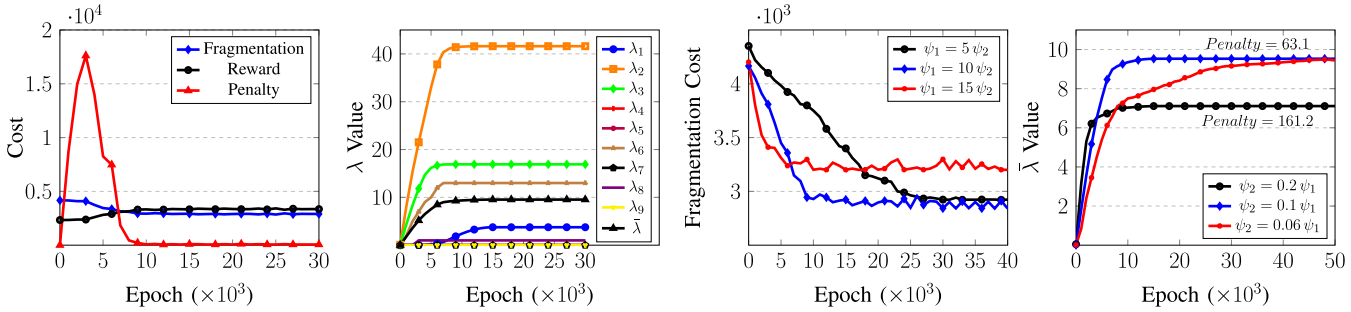
As in the previous experimentation we monitor various cost metrics during the training phase of the NCO agent, including the reward, and depict them in Fig. 6a. Notably, the fragmentation cost shows a rapid decrease and convergence after the initially high values. This indicates the agent's ability to quickly learn the placement policy that results in the lowest resource fragmentation. The reward follows the opposite path as it increases when the fragmentation cost is reduced. At the same time, the penalty cost starts low due to the zeroed Lagrange multipliers and rapidly increases after the first updates, guided by Eq. (29). Despite this initial surge, it quickly converges to zero as the agent learns to avoid violating the system's constraints effectively.

Additionally, in Fig. 6b we give some insights of the progression and convergence of the nine Lagrange multipliers that correspond to the system's constraints, Eqs. (8)– (16), over the course of training. A higher value indicates that the corresponding constraint is critical for the system and the optimization is sensitive to it. On the contrary, a lower value suggests that the constraint does not affect the optimization and barely limits the solution. Overall, this analysis demonstrates the NCO-RCPO agent's capacity to in-depth learning and consistently applying a cost-effective VNF placement strategy where the learned Lagrange multipliers effectively guide the agent toward policy convergence, striking a balance between minimizing resource fragmentation and adhering to the system's constraints.

Finally, in Figs. 6c and 6d we quantify the impact the learning rates have on the fragmentation cost and the mean Lagrange multipliers value respectively. A first observation is that when the fast timescale ψ_1 becomes even faster, the agent converges earlier. However, if the learning rate is too high, the agent overshoots and converges to a higher fragmentation cost. A similar behavior is observed for the slow timescale, where a lower ψ_2 means that the agent takes more time to converge and the mean Lagrange multipliers are higher, which subsequently makes the penalty term dominant compared to the fragmentation cost in the optimization. This leads to even stricter constraint satisfaction, i.e., a lower average penalty, but may have a negative impact on the fragmentation cost.

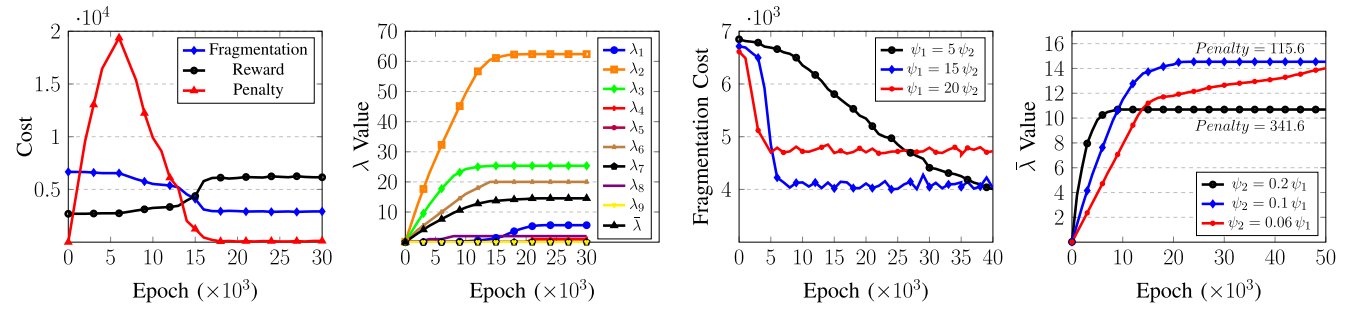
2) METROPOLITAN NETWORK

The training evolution of the NCO-RCPO agent for this topology is captured in Fig. 7a. The figure tracks the reward and the various costs similar to the previous setup, providing a clear picture of the agent's adaptation to a more intricate network. Additionally, Fig. 7b offers an insight into the evolution of the Lagrange multipliers, showing how the agent internally adjusts its penalty parameters in response to the more diverse metropolitan topology. The ψ_1 and ψ_2 impact analysis in Figs. 7c and 7d reveal similar patterns as in the



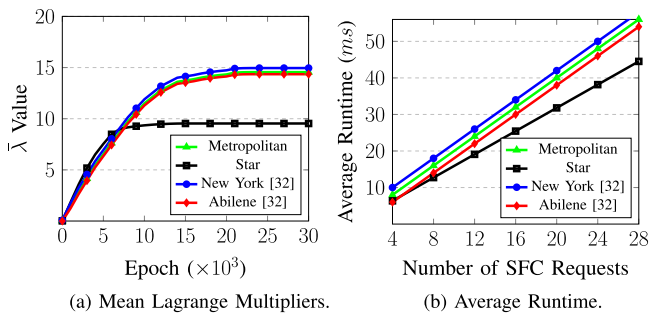
(a) Costs. (b) Lagrange Multipliers Values. (c) ψ_1 Impact. (d) ψ_2 Impact.

FIGURE 6. **Star network evaluation.**



(a) Costs. (b) Lagrange Multipliers Values. (c) ψ_1 Impact. (d) ψ_2 Impact.

FIGURE 7. **Metropolitan network evaluation.**



(a) Mean Lagrange Multipliers. (b) Average Runtime.

FIGURE 8. **Benchmarking across network topologies.**

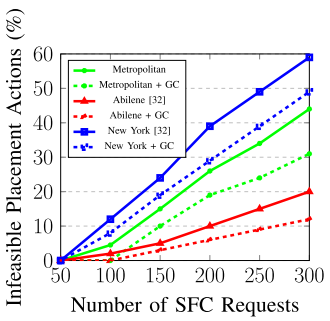


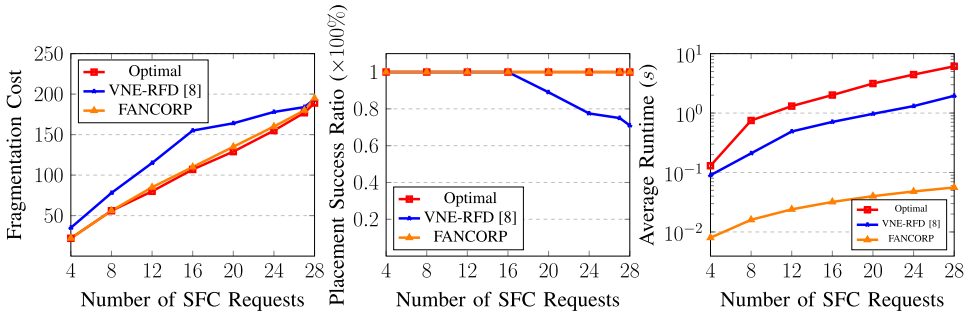
FIGURE 9. **GC Impact.**

Star network experiment, only this time the fragmentation cost and the mean Lagrange multiplier values are higher, as expected. Overall, we observe that in the Metropolitan network, the agent takes almost twice the time to converge to a placement policy that minimizes the costs, and these costs are higher.

Adding to that, the multipliers exhibit a slower increase and require more epochs to stabilize in complex topologies compared to the Star topology. This is clearly evinced in Fig. 8a, where we have isolated the mean values of the Lagrange multipliers for all four infrastructures: Metropolitan, Star, New York, and Abilene. A larger average multiplier value reflects that when the infrastructure complexity increases so does the need for a detailed penalizing approach to minimize resource fragmentation under the given constraints. Finally, a more complex network has a toll on the average runtime, as depicted in Fig. 8. Placing a batch of requested SFCs becomes increasingly slower as the batch size becomes larger, and this is mainly attributed to Dijkstra's algorithm which struggles to find available paths to connect the servers selected by the agent for placing the VNFs.

D. GREEDY CORRECTION EFFICIENCY

In every experimentation family so far, we observe that the constraint violation penalty sharply rises to its peak, then experiences a rapid decline and plateaus at the convergence



(a) Fragmentation Cost. (b) Placement Success Ratio. (c) Average Runtime.

FIGURE 10. FANCORP vs. VNE-RFD [8] vs. optimal.

very close to zero, oscillating slightly afterward. This slight oscillation could potentially result in placement errors from the agent, as discussed in Section IV-D. In this subsection, we show experimentally why it is important to complement the outcome of the NCO agent with the GC correction mechanism. We first remind the reader that the FANCORP agent has been trained to place up to 100 SFC requests. This training target is customizable and reflects the network operators' estimations of the anticipated load, providing a flexible basis for optimization to meet varying network demands. Out of these 100 SFC requests, the agent has managed to place 95 of them successfully on average in the Metropolitan network topology; the rest of the placement actions resulted in some constraints being violated, which deemed them initially infeasible. However, with GC's intervention, these actions were corrected, bringing this number to 100 on average, i.e., zero request rejections.

To amplify the effect of GC's corrective decisions, in the following experiment, we proceed to request the placement of more than 100 SFCs in three different network topologies: Metropolitan, Abilene, and New York. This is the point at which we expect the agent to start making significant miscalculations. Fig. 9 visually represents the placement error percentage in relation to the number of SFCs requested for these topologies. Solid lines represent the infeasible placement percentage without the Greedy Correction mechanism, whereas the dashed lines represent the infeasible placement percentage with GC. We observe that the dashed lines consistently show a lower percentage of i) VNE-RFD [8], which first introduced the RFD concept and then solved the fragmentation-aware VNF-CPP using a proactive and a reactive online heuristic algorithm, and ii) the optimal solution of the ILP formulation, which is once more computed through the Gurobi solver.uccessful SFC placements for up to 200 requests, which increases to a 13% improvement for 300 and more requests. GC's impact in improving robustness is evident. Similar improvements are reported for the other two networks, with the New York one having significantly more infeasible placements due to complex node connections and the Abilene network significantly less than the Metropolitan network due to simpler structure.

E. BENCHMARKING

For the last part of this evaluation, we benchmark the end-to-end operation of the proposed FANCORP framework on the Metropolitan network against two baselines: i) VNE-RFD [8], which first introduced the RFD concept and then solved the fragmentation aware VNF-CPP using a proactive and a reactive, online heuristic algorithm and ii) the optimal solution of the ILP formulation which is once more computed through the Gurobi solver. This analysis considers both the fragmentation cost, as defined in Eq. (6), as well as the SFC placement success ratio of up to 28 SFC requests. The results of the former are depicted in Fig. 10a. Naturally, as the number of SFC requests increases, all methods incur higher fragmentation costs due to the growing decision complexity in a resource-limited infrastructure. However, FANCORP performs close to the optimal ILP solution results, outperforming VNE-RFD. In Fig. 10b We depicted the average SFC placement success ratio in the same experiment. We observe that VNE-RFD method starts rejecting placements when more than 16 SFCs are requested. On the other hand, FANCORP again performs near-optimally and manages to successfully place all of the requested SFCs. This is due to the fact that the proposed framework's NCO-RCPO component dynamically adjusts the Lagrange multipliers during training, allowing the agent to adapt to varying network states and decisions that would lead to constraint violations and, eventually, placement failures. This adaptability rooted in RL techniques, together with GC's aid, ensures FANCORP's robust performance and represents a significant advancement over static heuristic methods. This also makes our framework suitable for real-time decision-making and increases its scalability, as depicted in Fig. 10c. In particular, from this figure, we observe that FANCORP places the batches of SFC requests in a fraction of the time required by the two other algorithms.

VI. CONCLUSION

In this work, we proposed a fragmentation-aware Neural Combinatorial Optimization with a constrained Policy Optimization (FANCORP) enabled framework for solving the VNF-CPP problem. This framework combines the

advantages of deep reinforcement learning with those of the reward-constrained policy optimization, enabling the adaptive learning of Lagrange multipliers that lead to the satisfaction of the system constraints under dynamic networking conditions. This contributes to a smoother and faster learning process, further enhancing the framework's efficiency in calculating SFC placement policies that minimize resource fragmentation in the infrastructure. The robustness of the proposed mechanism is enhanced by a Greedy Correction mechanism, a complementary heuristic algorithm that validates the feasibility of the produced solution and takes corrective actions in case of minor miscalculations. Our experimental study showcased the superiority of the FANCORP framework in terms of minimizing resource fragmentation, the average placement cost and the constraint violation degrees while maximizing the SFC placement success ratio when compared to state-of-the-art methods. In future work, we plan to extend the FANCORP framework to address more complex network scenarios and constraints and explore integrating a transformer model in the S2S place to improve its performance. Another promising direction we have pinpointed is investigating the potential of transfer learning and meta-learning approaches to enhance the NCO-RCPO agent's adaptability to network dynamicity.

REFERENCES

[1] T. Gao et al., "Cost-efficient VNF placement and scheduling in public cloud networks," *IEEE Trans. Commun.*, vol. 68, no. 8, pp. 4946–4959, Aug. 2020.

[2] G. L. Santos et al., "Service function chain placement in distributed scenarios: A systematic review," *J. Netw. Syst. Manage.*, vol. 30, no. 1, p. 4, Jan. 2022.

[3] W. Attaoui, E. Sabir, H. Elbiaze, and M. Guizani, "VNF and CNF placement in 5G: Recent advances and future trends," *IEEE Trans. Netw. Service Manage.*, vol. 1, no. 2, pp. 1–16, Jul. 2023.

[4] D. Qi, S. Shen, and G. Wang, "Towards an efficient VNF placement in network function virtualization," *Comput. Commun.*, vol. 138, pp. 1–28, Apr. 2019.

[5] A. Leivadeas, G. Kesidis, M. Ibnkahla, and I. Lambadaris, "VNF placement optimization at the edge and cloud," *Future Internet*, vol. 11, no. 3, p. 69, 2019.

[6] S. Wang, C. Yuen, W. Ni, Y. L. Guan, and T. Lv, "Multiagent deep reinforcement learning for cost- and delay-sensitive virtual network function placement and routing," *IEEE Trans. Commun.*, vol. 70, no. 8, pp. 5208–5224, Aug. 2022.

[7] N. He et al., "Leveraging deep reinforcement learning with attention mechanism for virtual network function placement and routing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 34, no. 4, pp. 1186–1201, Apr. 2023.

[8] H. Lu and F. Zhang, "Resource fragmentation-aware embedding in dynamic network virtualization environments," *IEEE Trans. Netw. Service Manage.*, vol. 19, no. 2, pp. 936–948, Jun. 2022.

[9] C. Tessler, D. J. Mankowitz, and S. Mannor, "Reward constrained policy optimization," 2018, *arXiv:1805.11074*.

[10] I. Bello, H. Pham, Q. V. Le, M. Norouzi, and S. Bengio, "Neural combinatorial optimization with reinforcement learning," 2016, *arXiv:1611.09940*.

[11] B. Németh, N. Molner, J. Martín-Pérez, C. J. Bernardos, A. de la Oliva, and B. Sonkoly, "Delay and reliability-constrained VNF placement on mobile and volatile 5G infrastructure," *IEEE Trans. Mobile Comput.*, vol. 21, no. 9, pp. 3150–3162, Sep. 2022.

[12] Y. Mao, X. Shang, and Y. Yang, "Joint resource management and flow scheduling for SFC deployment in hybrid edge-and-cloud network," in *Proc. IEEE Conf. Comput. Commun.*, Jul. 2022, pp. 170–179.

[13] V. Tran, J. Sun, B. Tang, and D. Pan, "Traffic-optimal virtual network function placement and migration in dynamic cloud data centers," in *Proc. IEEE Int. Parallel Distrib. Process. Symp. (IPDPS)*, Jul. 2022, pp. 919–929.

[14] Q. Zhang, F. Liu, and C. Zeng, "Online adaptive interference-aware VNF deployment and migration for 5G network slice," *IEEE/ACM Trans. Netw.*, vol. 29, no. 5, pp. 2115–2128, Oct. 2021.

[15] N. Promwongsa, A. Ebrahimzadeh, R. H. Glitho, and N. Crespi, "Joint VNF placement and scheduling for latency-sensitive services," *IEEE Trans. Netw. Sci. Eng.*, vol. 9, no. 4, pp. 2432–2449, May 2022.

[16] M. Bagaa, D. L. C. Dutra, T. Taleb, and H. Flinck, "Toward enabling network slice mobility to support 6G system," *IEEE Trans. Wireless Commun.*, vol. 21, no. 12, pp. 10130–10144, Dec. 2022.

[17] R. Solozabal, J. Ceberio, A. Sanchoyerto, L. Zabala, B. Blanco, and F. Liberal, "Virtual network function placement optimization with deep reinforcement learning," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 2, pp. 292–303, Feb. 2020.

[18] C. R. de Mendoza, B. Bakhshi, E. Zeydan, and J. Mangues-Bafalluy, "Near optimal VNF placement in edge-enabled 6G networks," in *Proc. 25th Conf. Innov. Clouds, Internet Netw. (ICIN)*, Mar. 2022, pp. 136–140.

[19] Z. Chen, H. Li, K. Ota, and M. Dong, "Deep reinforcement learning for AoI aware VNF placement in multiple source systems," in *Proc. IEEE Global Commun. Conf.*, Dec. 2022, pp. 2873–2878.

[20] L. A. Grieco, G. Boggia, G. Piro, Y. Jararweh, and C. Campolo, "Ad-hoc, mobile, and wireless networks," in *Proc. 19th Int. Conf. Ad-Hoc Netw. Wireless*, 2020, pp. 1–20.

[21] F. Bari, S. R. Chowdhury, R. Ahmed, R. Boutaba, and O. C. M. B. Duarte, "Orchestrating virtualized network functions," *IEEE Trans. Netw. Service Manage.*, vol. 13, no. 4, pp. 725–739, Dec. 2016.

[22] J. Fu and G. Li, "An efficient VNF deployment scheme for cloud networks," in *Proc. IEEE Int. Conf. Commun. Softw. Netw. (ICCSN)*, Jun. 2019, pp. 497–502.

[23] B. Yi, X. Wang, and M. Huang, "A generalized VNF sharing approach for service scheduling," *IEEE Commun. Lett.*, vol. 22, no. 1, pp. 73–76, Jan. 2018.

[24] H. Guo et al., "Cost-aware placement and chaining of service function chain with VNF instance sharing," in *Proc. IEEE/IFIP Netw. Oper. Manage. Symp.*, Apr. 2020, pp. 1–8.

[25] R. Mohamed, M. Avgeris, A. Leivadeas, and I. Lambadaris, "Fragmentation-aware VNF placement: A deep reinforcement learning approach," in *Proc. IEEE Int. Conf. Commun. (ICC)*, vol. 135, Jun. 2024, pp. 5257–5262.

[26] J. Halpern and C. Pignataro, *Service Function Chaining (SFC) Architecture*, document RFC 7665, 2015.

[27] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," 2014, *arXiv:1409.0473*.

[28] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Reinforcement Learn.*, vol. 8, pp. 5–32, May 1992.

[29] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*, vol. 135. Cambridge, MA, USA: MIT Press, 1998.

[30] H. Ma, C. Liu, S. E. Li, S. Zheng, W. Sun, and J. Chen, "Learn zero-constraint-Violation safe policy in model-free constrained reinforcement learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 2, no. 2, pp. 1–15, Jul. 2024.

[31] R. Mohamed et al., "Service function chain network planning through offline, online and infeasibility restoration techniques," *Comput. Netw.*, vol. 242, Apr. 2024, Art. no. 110241.

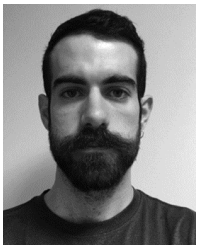
[32] S. Orlowski, R. Wessäly, M. Pióro, and A. Tomaszewski, "SNDlib 1.0—Survivable network design library," *Netw., Int. J.*, vol. 55, no. 3, pp. 276–286, 2010.

[33] *IBM Cloud Docs*. Accessed: Jun. 30, 2024. [Online]. Available: https://cloud.ibm.com/docs/bare-metal?topic=bare-metal-about-bm

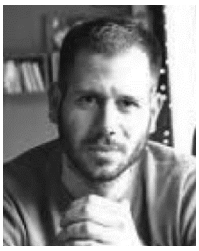
[34] G. Papathanail, I. Sakellariou, L. Mamatas, and P. Papadimitriou, "Dynamic schedule computation for time-aware shaper in converged IoT-cloud environments," in *Proc. 27th Conf. Innov. Clouds, Internet Netw. (ICIN)*, vol. 16, Mar. 2024, pp. 1–8.

[35] D. M. Manias, I. Shaer, J. Naoum-Sawaya, and A. Shami, "Robust and reliable SFC placement in resource-constrained multi-tenant MEC-enabled networks," *IEEE Trans. Netw. Service Manage.*, vol. 1, no. 1, pp. 1–19, Jun. 2023.

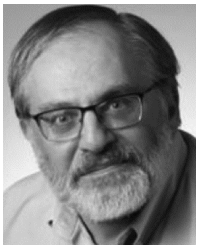
RAMY MOHAMED (Member, IEEE) was born in Alexandria, Egypt. He received the bachelor's degree in electronics and communications engineering from the Arab Academy for Science, Technology and Maritime Transport, Aswan, Egypt, in 2014, and the Master of Science degree in electrical, electronics, and communications engineering from Aswan University, Egypt, in 2018. He is currently pursuing the Ph.D. degree with the Systems and Computer Engineering Department, Carleton University, Ottawa, ON, Canada. He is also a Teaching Assistant and a Research Assistant with the Systems and Computer Engineering Department, Carleton University. He is a Certified Cloud Systems Specialist with CENGN, with more than eight years of professional and research experience. His research interests include cloud computing systems, 5G/6G technologies, and machine learning applications in telecommunications. He received several awards and honors, including the Queen Elizabeth II Graduate Scholarship in Science and Technology (QEII-GSST), the Dr. Roger Kaye Memorial Scholarship, and Ontario Graduate Scholarship (OGS).



MARIOS AVGERIS (Member, IEEE) received the Diploma degree from the Department of Electrical and Computer Engineering (ECE), National Technical University of Athens (NTUA), Greece, in 2016, and the Ph.D. degree in dynamic resource allocation and computational offloading from the Network Edge for Internet of Things Applications, in 2021. He is currently a full-time Postdoctoral Fellow with Carleton University, Ottawa, Canada. He is also affiliated with École de technologie supérieure (ÉTS) and Ericsson. His research interests include control theory, reinforcement learning, network virtualization, computational offloading, edge and cloud computing, and the IoT. He has been awarded with the 2023 CU-PSAC Postdoctoral Fellow Research Award.



ARIS LEIVADEAS (Senior Member, IEEE) received the Diploma degree in electrical and computer engineering from the University of Patras, in 2008, the M.Sc. degree in engineering from King's College London, in 2009, and the Ph.D. degree in electrical and computer engineering from the National Technical University of Athens, in 2015. From 2015 to 2018, he was a Postdoctoral Fellow with the Department of Systems and Computer Engineering, Carleton University, Ottawa, Canada. In parallel, he was with Ericsson and later collaborated with Cisco, Ottawa. He is currently an Associate Professor with the Department of Software and Information Technology Engineering, École de technologie supérieure, Montreal, Canada. His research interests include network function virtualization, cloud and edge computing, the IoT, and network optimization and management. He received the best paper awards in ACM ICPE'18, ACM ICPE'23, and the IEEE iThings'21 and the Best Presentation Award in IEEE HPSR'20.



IOANNIS LAMBADARIS (Senior Member, IEEE) was born in Thessaloniki, Greece. He received the Diploma degree in electrical engineering from the Polytechnic School, Aristotle University of Thessaloniki, Thessaloniki, in 1984, the M.Sc. degree in engineering from Brown University, Providence, RI, USA, in 1985, and the Ph.D. degree in electrical engineering from the Department of Electrical Engineering, Systems Research Center (SRC), Institute for Systems Research (ISR), University of Maryland, College Park, MD, USA, in 1991. After finishing his graduate education, he was a Research Associate with Concordia University, Montreal, QC, Canada, from 1991 to 1992. Since September 1992, he has been with the Department of Systems and Computer Engineering, Carleton University, Ottawa, ON, Canada, where he is currently a Professor. His interests lie in the area of applied stochastic processes and their application for modeling and performance analysis of computer communication networks and wireless networks. His current research interests include quality of service (QoS) control for IP and evolving optical networks architectures and stochastic control/optimization in emerging wireless networks. His research is done in close collaboration with his students and colleagues in the Broadband Networks Laboratory. He received a fellowship from the National Fellowship Foundation of Greece (1980–1984) during his undergraduate studies. He also received the Technical Chamber of Greece Award (ranked first in graduating class). He received a Fulbright Fellowship (1984–1985) for graduate studies in USA. While at Carleton University, he received the Premiers Research Excellence Award, and the Carleton University Research Excellence Award (2000–2001), for his research achievements in the area of modeling and performance analysis of computer networks. Since 2022, he has been the Ericsson Chair of the 5G Wireless Research, Carleton University.