Computing&AI Connect
ISSN: 3006-4163
2025, Article ID. 2025.0012, Cite as: https://doi.org/10.69709/CAIC.2025.181989

SCIFINITI

Research Article

OPEN ACCESS

A Novel Transformer Reinforcement Learning-Based NFV Service Placement in MEC Networks

Abdoulaye Ndiaye[✉,1,2,*] **Mouhamad Dieye**[✉,3,*] **Wael Jaafar**[✉,2,*] **Fatoumata Balde**[✉,1] **Roch Glitho**[✉,4]

[1] Université Alioune Diop de Bambey, Bambey P.O. Box 30, Sénégal
[2] Software and IT Engineering Department, École de Technologie Supérieure, Montréal, QC H3C 1K3, Canada
[3] CIISE, Concordia University, Montréal, QC H3G 1M8, Canada
[4] University of Western Cape, Capetown 7535, South Africa

Article History
Submitted: September 10, 2024 Accepted: March 24, 2025 Published: May 07, 2025

Abstract
The advent of 5G networks has facilitated various Industry 4.0 applications requiring stringent Quality-of-Service (QoS) demands, notably Ultra-Reliable Low-Latency Communication (URLLC). Multi-Access Edge Computing (MEC) has emerged as a key technology to support these URLLC applications by bringing computational resources closer to the user, thus reducing latency. Meanwhile, Network Function Virtualization (NFV) supports 5G networks by offering flexibility and scalability in service provisioning across various applications. Despite their benefits, MEC networks must adapt to dynamically fluctuating user demands and varying workloads, which can create challenges in maintaining QoS. This paper addresses the Virtual Network Function (VNF) placement problem in MEC networks, focusing on minimizing costs while ensuring QoS through VNF reuse. We propose a novel solution based on the Deep Transformer Q-network (DTQN) algorithm, leveraging reinforcement learning to optimize VNF placement and redeployment. Extensive simulations demonstrate that our DTQN algorithm outperforms baseline approaches, achieving up to a 9% improvement over the D3T-based method and up to 56% over the DQN-based method in terms of average rewards under specific scenarios. This results in significant improvements in cost efficiency, resource utilization, and QoS maintenance.

Keywords:
network function placement; network function chaining; deep transformer Q-network (DTQN); reinforcement learning; edge computing; service migration.

1. Introduction

The advent of the 5th generation (5G) networks has facilitated various Industry 4.0 applications, including connected and autonomous vehicles, remote surgery, industrial automation, and mission-critical Internet-of-Things (IoT) services. These applications, classified as Ultra-Reliable Low-Latency Communication (URLLC) in 5G, demand stringent Quality-of-Service (QoS) requirements, notably low latency and high reliability [1].

Two key technologies have emerged to support the flexible and cost-effective provisioning of applications while ensuring QoS requirements, namely Network Function Virtualization (NFV) and Multi-Access Edge Computing (MEC). On the one hand, NFV has become fundamental for service provisioning, enabling software-defined virtual networks to offer flexibility and scalability. Delivering a specific service requires composing and mapping an ordered chain of Virtual Network Functions (VNFs) to the underlying network infrastructure [2]. On the other hand, MEC alleviates resource demands and reduces user-

* **Corresponding Authors:**
Abdoulaye Ndiaye, Université Alioune Diop de Bambey, Bambey P.O. Box 30, Sénégal; Software and IT Engineering Department, École de Technologie Supérieure, Montréal, QC H3C 1K3, Canada, abdoulaye.ndiaye.4@ens.etsmtl.ca; Mouhamad Dieye, CIISE, Concordia University, Montréal, QC H3G 1M8, Canada, mouhamad.dieye@concordia.ca; Wael Jaafar, Software and IT Engineering Department, École de Technologie Supérieure, Montréal, QC H3C 1K3, Canada, wael.jaafar@etsmtl.ca

perceived latency by extending cloud computing capabilities and IT services to the edge of the network [3]. Despite their advantages, MEC nodes possess comparatively limited storage and computational resources compared to the cloud.

In this context, a significant challenge in deploying VNFs for URLLC applications in MEC networks arises from the dynamic nature of user demands, application requirements, and traffic conditions, which may lead to several violations of QoS requirements. Furthermore, the typically ephemeral nature of VNFs means that deployed instances become inactive or must be deactivated once the related service is completed, thereby incurring additional costs or leading to inefficient resource usage [4].

In the literature, typical solutions for NFV deployment in MEC networks involve modifying the deployed VNFs through various redeployment schemes, including VNF migration, VNF sharing, and scaling [5–10]. VNF migration is frequently used to achieve load balancing in congested networks, conserve energy in underutilized segments, ensure reliability during physical node failures, and minimize costs. However, VNF migration has to guarantee stringent latency requirements between VNFs within a VNF chain. Traditional migration strategies typically move a single VNF at a time, which incurs high transmission costs and introduces potential service disruptions due to the resource overhead required for migration, such as processing power and bandwidth to transfer VNF states. These inefficiencies can degrade the user experience, strain network resources, and potentially lead to service-level agreement (SLA) violations [11]. To mitigate these drawbacks, recent research by Afrasiabi et al. [12] has proposed cluster migration, where groups of coupled VNFs are migrated together within a single physical node or across multiple nodes. This approach addresses some inefficiencies associated with single VNF migrations, yet the challenge of maintaining reliable service delivery remains a significant concern. Additionally, frequent migrations, whether from single VNFs or clusters, can lead to resource fragmentation, further reducing overall network efficiency and complicating resource management.

In contrast, reusing VNFs ensures consistent service delivery, minimizes the risk of interruptions, reduces downtime, and eliminates the need to migrate VNF states and data across nodes. It leverages tested and stable configurations to enhance the reliability and reduce the complexity of fault management. VNF reuse is cost-efficient as it avoids migration-related transmission and operational costs, maintains low latency by keeping VNFs in optimal locations, and enhances resource utilization. However, only a few studies explored reuse schemes [4,13,14], and generally overlooked the impact of traffic redirection on the operation and reliability of Service Function Chains (SFCs). Moreover, most existing work proposed heuristic algorithms that may require long execution times, yielding only suboptimal solutions and facing exponential computational complexity with network scaling.

Consequently, this paper focuses on solving the problem of SFC/VNF placement in MEC networks for mission-critical applications, aiming to guarantee QoS requirements through VNF redeployment and reuse schemes. Given that the SFC/VNF placement problem is NP-hard [15–17], the complexity of our problem, which includes the placement, recomposition, and redeployment of SFC/VNF, is also NP-hard. To address it, we formulate the problem as a Markov Decision Process (MDP) and solve it using a novel reinforcement learning (RL)-based framework. Specifically, we propose a novel and intelligent VNF/SFC placement strategy based on the deep Transformer Q-network (DTQN) algorithm. This approach is designed to learn complex data representations, enabling accurate and efficient decision-making. In particular, we aim to minimize the costs associated with VNF placement/redeployment (e.g., migration, removal) while favoring VNF reuse to reduce service disruptions and satisfy QoS requirements.

Our contributions can be summarized as follows:

1. We formulate the VNF/SFC placement problem in MEC networks for mission-critical applications using Mixed-Integer Non-Linear Programming (MINLP).
2. Given the NP-hardness of the formulated problem, we propose a novel DTQN-based approach to solve it.
3. We perform extensive simulations that demonstrate the efficiency of our solution in terms of cost, VNF placement, reuse, and migration of VNF, as compared to the baseline approaches.

The remainder of the paper is organized as follows: Section 2 discusses the related work. In Section 3, we present the system model, while Section 4 formulates the VNF placement problem. Section 5 details our proposed solution, and Section 6 provides the simulation results. Finally, Section 7 concludes the paper.

2. Related Work

To overcome the critical challenges of URLLC application provisioning in MEC networks, it is essential to meet key requirements such as minimizing latency to fulfill stringent QoS demands, reducing operational costs to en-

2025, Vol. 2, Article ID. 2025.0012
https://doi.org/10.69709/CAIC.2025.181989

sure scalability, and maintaining the reliability of services through effective VNF reuse and minimal service disruptions. Comprehensive overviews of NFV placement strategies within MEC networks have been provided in surveys [18–21].

In the context of latency minimization and cost efficiency, the advent of NFV and MEC as fundamental technologies for agile network service deployment in 5G has led to extensive research. Early studies primarily employed heuristic and optimization-based approaches to address these challenges. For example, Yala et al. [22] used a genetic algorithm to balance the conflicting objectives of minimizing latency and maximizing availability in NFV-MEC networks. Kiran et al. [14] focused on reducing placement and resource costs through coordinated VNF placement, demonstrating its effectiveness in lowering overall costs. Jin et al. [23] addressed VNF chain deployment in MEC environments by formulating the problem as a Mixed Integer Linear Programming (MILP) model to optimize resource consumption. They proposed a two-phase deployment scheme: a constrained depth-first search algorithm to identify feasible paths and a path-based greedy algorithm to maximize resource reuse while minimizing new allocations. Simulations revealed that this method achieved near-optimal performance, reducing resource consumption by up to 25.6% compared to heuristic baselines while meeting latency constraints. Similarly, Afrasiabi et al. [12] proposed clustering coupled VNFs and migrating them within or across physical nodes using two look-ahead heuristics to reduce embedding costs and avoid local optima. Despite their merits, these approaches do not address reliability issues comprehensively.

The increasing complexity and dynamism of network environments necessitate scalable and adaptive solutions. To sustain service performance, researchers have widely studied reconfiguration-based strategies, such as VNF migration. Li et al. [11] introduced a latency-aware migration strategy to reduce end-to-end latency in SFC deployments, while Cho et al. [24] investigated optimization techniques for VNF migration in dynamic cloud environments. Kuo et al. [25] proposed a joint VNF placement and path selection approach to maximize served traffic demands, using a stress-testing-inspired algorithm to adapt resource allocation dynamically. This method demonstrated superior performance in terms of link and Virtual Machine (VM) resource utilization, outperforming traditional heuristics. However, scalability remains a challenge, as many of these solutions struggle to maintain optimal performance in highly dynamic conditions. Zhu et al. [26] similarly noted that while these strategies can be effective, they often fall short in environments requiring rapid adaptation to changing conditions.

An alternative strategy to enhance reliability and reduce frequent redeployments or migrations involves reusing VNFs. Addressing this gap, Doan et al. [27] proposed the Subchain-Aware NFV Service Placement (SAP) model, which optimizes network function reuse in MEC environments. By focusing on subchain reuse, SAP reduces configuration and deployment costs while enhancing reliability. The authors developed Tabu-SAP, a Tabu search-based solution, and implemented the Automated Provisioning framework for MEC (APMEC) on OpenStack. Tabu-SAP demonstrated scalability and supported eight times more SFC requests, achieving over a 50 percent cost reduction compared to start-of-the-art methods.

To address the limitations of traditional approaches, recent studies have incorporated machine learning (ML) and RL methods, which offer improved adaptability and scalability. For instance, Abouaomar et al. [28] applied deep reinforcement learning to optimize service migration in MEC vehicular environments, minimizing latency and service disruptions. Similarly, Subramanya et al. [29] utilized neural networks to enable auto-scaling of VNFs based on traffic demand and latency requirements. While these approaches show promise, traditional ML methods often face challenges in stability, scalability, and efficiently capturing long-term dependencies in dynamic settings.

Transformer-based techniques have emerged as a promising solution to address these challenges. Wu et al. [30] introduced the Double Deep Q-Network Decision Transformer (D3T), which combines a Decision Transformer (DT) with a Double Deep Q-Network (DDQN). The DDQN generates trajectory data for offline training, while the DT models sequences to optimize placement decisions, effectively handling high-dimensional state-action distributions and mitigating issues such as error propagation and value overestimation. D3T achieved a 25% reduction in rejection ratio and a 7% reduction in delay compared to previous methods. However, it does not fully address reliability or redeployment strategies to ensure continuous service delivery.

In summary, the existing body of research on NFV and MEC highlights substantial progress in optimizing service deployment for 5G applications, particularly in latency-sensitive and cost-efficient scenarios. Early heuristic and optimization-based approaches provided foundational insights but struggled with scalability and adaptability to dynamic environments. More recent reinforcement learning methods have improved decision-making in

such dynamic environments, yet often fail to fully address temporal dependencies and the complexity of large-scale, mission-critical applications. Transformer-based models have shown promise in overcoming these limitations, offering superior capabilities for handling high-dimensional and temporally dependent data. In this context, our work advances the state of the art by introducing a novel DTQN-based framework that leverages the strengths of transformers and reinforcement learning. This approach uniquely focuses on optimizing VNF placement through reuse and intelligent redeployment, addressing critical gaps in scalability, cost efficiency, reliability, and QoS adherence.

3. System Model

We consider a two-tier hierarchical MEC network model. The lowest tier consists of user devices such as sensors, smart gadgets, and cameras that generate and transmit real-time data. This data is forwarded to edge nodes in the above MEC layer for processing. Indeed, the MEC nodes can compute tasks, store data, and network with other nodes.

We assume that time is discretized into periods $\mathcal{T} = \{1, 2, \ldots, T\}$, with each period denoted by $t \in \mathcal{T}$. At the start of each period, placement decisions are made to ensure the provision of NFV-based services within predefined QoS requirements. The system must decide, for each computing request, whether to utilize existing VNF instances or create new ones at time t. Reusing instances incur a reuse cost, which is generally lower than the instantiation cost of a new VNF instance. The length of each period strikes a balance between the system's responsiveness to dynamic changes in user demands and network conditions and the computational overhead incurred.

The physical network is modeled as a directed graph $\mathcal{G} = (\mathcal{N}, \mathcal{L})$, where $\mathcal{N}$ represents the physical MEC nodes and $\mathcal{L}$ represents the physical links. Each node $n \in \mathcal{N}$ is characterized by the tuple $\{c_n, \mathcal{C}_n, \alpha_n\}$, where c_n represents the maximum computing resource capacity, $\mathcal{C}_n$ the unit resource cost, and α_n, its reliability. Each link $l \in \mathcal{L}$ is characterized by the tuple $\{b_l, d_l, \mathcal{C}_l, \alpha_l\}$, where b_l denotes the available bandwidth capacity, d_l indicates the link delay, $\mathcal{C}_l$ represents the unit bandwidth cost, and α_l represents the reliability of the link l.

Let $\mathcal{K}$ be the set of available VNF types. Assume a licensing model where a maximum number I^k of VNF instances may be instantiated across the network for VNF k, with $i \in \{1, 2, \ldots, I^k\}$. Each VNF type $k \in \mathcal{K}$ is characterized by the tuple $\{q^k, u^k, c^k, \mathcal{C}_n^k, \mathcal{C}_n^{i,k}, \alpha^k\}$, where q^k represents the processing demand, u^k and c^k represent the resource and processing capacities, $\mathcal{C}_n^k$ is the instantiation cost for creating a new instance on node n, $\mathcal{C}_n^{i,k}$ is the cost of reusing instance i on node n, and α^k represents the reliability of VNF k.

Define the binary matrix $\mathbf{X}(t) \in \{0,1\}^{|\mathcal{N}|\times|\mathcal{K}|\times I^k}$ to indicate whether a new instance i of VNF k is deployed on node n at time t. Let the binary matrix $\mathbf{M}(t) \in \{0,1\}^{|\mathcal{N}|\times|\mathcal{K}|\times I^k}$ indicate whether instance i of VNF k is reused on node n at time t or not, and $\mathbf{\Gamma} \in \mathbb{R}^{|\mathcal{N}|\times|\mathcal{K}|\times I^k}$ denotes the processing load already assigned to instance i of VNF k on node n.

Moreover, let $\mathcal{R}$ be the set of requests. A request $r \in \mathcal{R}$ is modeled as a directed graph $\mathcal{G}_r = (\mathcal{K}_r, \mathcal{L}_r)$, where $\mathcal{K}_r$ is the set of VNFs to be installed on MEC nodes and $\mathcal{L}_r$ is the set of virtual edges. A request r is characterized by its delay threshold d_r and reliability requirement α_r. Each virtual edge $l_r \in \mathcal{L}_r$ is characterized by a tuple $\{d_{l,r}, b_{l,r}\}$, where $d_{l,r}$ and $b_{l,r}$ represent the delay and bandwidth requirement, respectively. Finally, the binary matrix $\mathbf{Y}(t) \in \{0,1\}^{|\mathcal{L}|\times|\mathcal{L}_r|}$ indicates whether a virtual link l_r is mapped to a physical link l at time t or not.

4. Problem Formulation

In this section, we first outline the constraints necessary for our problem formulation, then provide detailed formulations of the associated costs, and finally formulate the objective function.

4.1. Constraints

Each instance i of VNF k cannot be deployed more than once in the network. Furthermore, the number of deployed instances for a given VNF k cannot exceed its maximum number of allowed instances I_k. Hence,

$$\sum_{n\in\mathcal{N}} \mathbf{X}_{n,k,i}(t) \leq 1, \quad \forall k \in \mathcal{K},\ i \in I_k,\ t \in \mathcal{T} \tag{1}$$

and

$$\sum_{n\in\mathcal{N}} \sum_{t\in\mathcal{T}} (\mathbf{X}_{n,k,i}(t) + \mathbf{M}_{n,k,i}(t)) \leq I^k, \quad \forall k \in \mathcal{K},\ i \in \{1, 2, \ldots, I^k\}. \tag{2}$$

At any given time t, for a specific instance i of VNF k on node n, either a new instantiation or reuse should be chosen, but not both simultaneously. In addition, reuse is only possible if the instance has already been created, i.e.,

$$\mathbf{X}_{n,k,i}(t) + \mathbf{M}_{n,k,i}(t) \leq 1, \quad \forall k \in \mathcal{K},\ i \in I_k,\ n \in \mathcal{N},\ t \in \mathcal{T}, \tag{3}$$

2025, Vol. 2, Article ID. 2025.0012
https://doi.org/10.69709/CAIC.2025.181989

and

$$\mathbf{M}_{n,k,i}(t) \leq \sum_{t'<t} \mathbf{X}_{n,k,i}(t'), \quad \forall k \in \mathcal{K},\ i \in I_k,\ n \in \mathcal{N},\ t, t' \in \mathcal{T}, \tag{4}$$

where t' represents a previous period. Constraint (4) can be linearized by introducing an auxiliary binary variable $\mathbf{Z}_{n,k,i}(t) = 1$ when a VNF k's instance i was instantiated at node n at $t' < t$ such that

$$\mathbf{Z}_{n,k,i}(t) \geq \mathbf{X}_{n,k,i}(t') \quad \forall t' < t, \forall n \in \mathcal{N}, \forall k \in \mathcal{K}, \forall i \in I^k, \tag{5}$$

and

$$\mathbf{M}_{n,k,i}(t) \leq \mathbf{Z}_{n,k,i}(t) \quad \forall n \in \mathcal{N}, \forall k \in \mathcal{K}, \forall i \in I^k, \forall t \in \mathcal{T}. \tag{6}$$

The processing load assigned to instance i of VNF k cannot exceed its available processing capacity, i.e.,

$$\sum_{n \in \mathcal{N}} q^k(\mathbf{X}_{n,k,i}(t) + \mathbf{M}_{n,k,i}(t)) \leq c^k - \Gamma_{n,k,i}, \quad \forall k \in \mathcal{K},\ i \in I_k,\ t \in \mathcal{T}. \tag{7}$$

Given the limited computing resources at node n, we must ensure that its total resource usage is not exceeded at any time as follows:

$$\sum_{k \in \mathcal{K}} \sum_{i=1}^{I^k} (\mathbf{X}_{n,k,i}(t) + \mathbf{M}_{n,k,i}(t)) \cdot u^k \leq c_n \quad \forall n \in \mathcal{N}, \forall t \in \mathcal{T}. \tag{8}$$

Also, the total bandwidth usage on physical link l should not exceed b_l at any time t, i.e.,

$$\sum_{r \in \mathcal{R}} \sum_{l_r \in \mathcal{L}_r} \mathbf{Y}_{l,l_r}(t) \cdot b_{l,r} \leq b_l \quad \forall l \in \mathcal{L}, \forall t \in \mathcal{T}, \tag{9}$$

while the delay requirement of each virtual link must be met,

$$\sum_{l \in \mathcal{L}} \mathbf{Y}_{l,l_r}(t) \cdot d_l \leq d_{l_r}, \quad \forall l_r \in \mathcal{L}_r. \tag{10}$$

Similarly, the delay requirement d_r for each request r should be met. In other words, the total delay of the path assigned to request r should not exceed d_r, as follows:

$$\sum_{l \in \mathcal{L}} \sum_{l_r \in \mathcal{L}_r} \mathbf{Y}_{l,l_r}(t) \cdot d_l \leq d_r \quad \forall r \in \mathcal{R}, \forall t \in \mathcal{T}. \tag{11}$$

Since VNFs can be virtually linked together to form a VNF chain or an SFC, the latter should have at least a physical link assigned between them, i.e.,

$$\mathbf{X}_{n,k,i}(t) \cdot \mathbf{X}_{n',k+1,j}(t) \leq \sum_{l \in \mathcal{L}} \mathbf{Y}_{l,l_r}(t), \quad \forall n, n' \in \mathcal{N}, k \in \mathcal{K}, l_r \in \mathcal{L}_r, n \neq n'. \tag{12}$$

Constraint (12) can be linearized by introducing an auxiliary binary variable $\mathbf{W}_{n,n',k,j,l_r}(t) \in \{0,1\}$ such that

$$\mathbf{W}_{n,n',k,j,l_r}(t) \leq \mathbf{X}_{n,k,i}(t) \quad \forall n, n' \in \mathcal{N}, \forall k \in \mathcal{K}, \forall l_r \in \mathcal{L}_r, \forall t \in \mathcal{T}, \tag{13}$$

$$\mathbf{W}_{n,n',k,j,l_r}(t) \leq \mathbf{X}_{n',k+1,j}(t) \quad \forall n, n' \in \mathcal{N}, \forall k \in \mathcal{K}, \forall l_r \in \mathcal{L}_r, \forall t \in \mathcal{T}, \tag{14}$$

$$\mathbf{W}_{n,n',k,j,l_r}(t) \geq \mathbf{X}_{n,k,i}(t) + \mathbf{X}_{n',k+1,j}(t) - 1 \quad \forall n, n' \in \mathcal{N}, \forall k \in \mathcal{K}, \forall l_r \in \mathcal{L}_r, \forall t \in \mathcal{T}, \tag{15}$$

and

$$\mathbf{W}_{n,n',k,j,l_r}(t) \leq \sum_{l \in \mathcal{L}} \mathbf{Y}_{l,l_r}(t) \quad \forall n, n' \in \mathcal{N}, \forall k \in \mathcal{K}, \forall l_r \in \mathcal{L}_r, \forall t \in \mathcal{T}. \tag{16}$$

Moreover, the reliability requirement α_r for each request $r \in \mathcal{R}$ should be met as follows:

$$\prod_{k \in \mathcal{K}_r} \prod_{i=1}^{I^k} (\mathbf{X}_{n,k,i}(t) \cdot \alpha^k + \mathbf{M}_{n,k,i}(t) \cdot \alpha^k) \cdot \prod_{l \in \mathcal{L}} \prod_{l_r \in \mathcal{L}_r} (\mathbf{Y}_{l,l_r}(t) \cdot \alpha_l) \geq \alpha_r,\ \forall r \in \mathcal{R}, \forall t \in \mathcal{T}. \tag{17}$$

For simplicity, we transform Constraint (17) into a logarithmic form, enabling its support by MINLP solvers. Specifically, we define the auxiliary matrix $\boldsymbol{\theta}(t) \in \mathbb{R}^{|\mathcal{N}| \times |\mathcal{K}| \times I^k}$ where $\theta_{n,k,i}(t)$ takes the value $\log(\alpha^k)$ if either $\mathbf{X}_{n,k,i}(t)$ or $\mathbf{M}_{n,k,i}(t)$ is equal to 1. Similarly, we define the auxiliary matrix $\boldsymbol{\phi}(t) = [\phi_{l,l_r}] \in \mathbb{R}^{|\mathcal{L}| \times |\mathcal{L}_r|}$ for the link mappings. Consequently, we obtain the following:

$$\theta_{n,k,i}(t) \geq (\mathbf{X}_{n,k,i}(t) + \mathbf{M}_{n,k,i}(t)) \cdot \log(\alpha^k), \quad \forall n \in \mathcal{N}, \forall k \in \mathcal{K}, \forall i \in I^k, \forall t \in \mathcal{T}, \tag{18}$$

$$\phi_{l,l_r}(t) \geq \mathbf{Y}_{l,l_r}(t) \cdot \log(\alpha_l), \quad \forall l \in \mathcal{L}, \forall l_r \in \mathcal{L}_r, \forall t \in \mathcal{T}, \tag{19}$$

and

$$\sum_{k \in \mathcal{K}_r} \sum_{i=1}^{I^k} \theta_{n,k,i}(t) + \sum_{l \in \mathcal{L}} \sum_{l_r \in \mathcal{L}_r} \phi_{l,l_r}(t) \geq \log(\alpha_r), \quad \forall r \in \mathcal{R}, \forall t \in \mathcal{T}. \tag{20}$$

4.2. Costs and Objective Function

The cost associated with the resource usage of MEC nodes is given by

$$\mathcal{C}_{res}(t) = \sum_{n \in \mathcal{N}} \sum_{k \in \mathcal{K}} \sum_{i=1}^{I^k} (\mathbf{X}_{n,k,i}(t) + \mathbf{M}_{n,k,i}(t)) \cdot u^k \cdot \mathcal{C}_n. \tag{21}$$

Also, we derive the costs related to instantiation and reuse of VNF instances as follows:

$$\mathcal{C}_{pla}(t) = \sum_{k\in\mathcal{K}} \sum_{i=1}^{I^k} \sum_{n\in\mathcal{N}} \left(\mathbf{X}_{n,k,i}(t) \cdot \mathcal{C}^k + \mathbf{M}_{n,k,i}(t) \cdot \mathcal{C}^{i,k}\right). \quad (22)$$

Moreover, we deduce the costs associated with the bandwidth usage of physical links by

$$\mathcal{C}_{bw}(t) = \sum_{l\in\mathcal{L}} \sum_{r\in\mathcal{R}} \sum_{l_r\in\mathcal{L}_r} \mathbf{Y}_{l,l_r}(t) \cdot b_{l,r} \cdot \mathcal{C}_l. \quad (23)$$

Hence, the total cost is defined by

$$\mathcal{C}_{tot}(t) = \mathcal{C}_{res}(t) + \mathcal{C}_{pla}(t) + \mathcal{C}_{bw}(t). \quad (24)$$

Finally, the objective of our optimization problem is formulated as

$$\min_{\substack{\mathbf{X}(t),\mathbf{M}(t),\mathbf{Y}(t),\\ \mathbf{Z}(t),\mathbf{W}(t),\theta(t),\phi(t)}} \sum_{t\in\mathcal{T}} \mathcal{C}_{tot}(t), \quad (25)$$

where the objective function aims to reduce the total cost of VNF deployment over the observation time $\mathcal{T}$. The resulting problem (P1), formed by equations (1)–(3), (5)–(11), (13)–(16), (18)–(20) and (25) is NP-hard. To prove this, we leverage the NP-hardness reduction approach by considering a simplified single-time slot problem where VNFs are placed in nodes to minimize costs while guaranteeing end-user QoS requirements. We further focus on the specific case of homogeneous nodes, which have the same capacity, while VNFs may have varying resource requirements. This particular problem can be easily understood as the well-known Generalized Assignment Problem (GAP) [31]. In GAP, the objective is to obtain a maximum overall profit assignment of tasks with different amounts of resources to agents, such that each task is assigned to precisely one agent, subject to the agents' capacity. Also, each task has a different profit, depending on the assigned agent. The defined problem can be viewed as a GAP, where agents correspond to nodes, VNFs represent tasks, and task profit is interpreted as cost savings. Since GAP is known to be NP-hard [31], then, by restriction, our simplified version of the problem is also NP-hard.

5. Proposed Solution

In this section, we propose a solution to the problem (P1), which addresses the total cost of SFC placement in MEC networks with redeployment schemes. We propose a DTQN-based approach that learns a policy for placing and routing network functions to ensure QoS requirements, considering real-time network conditions and operational objectives.

5.1. Background

Deep neural networks, particularly DQN, form the computational backbone of RL-based approaches, enabling robust performance across various domains [32]. Within the context of NFV, DQN significantly enhances the decision-making process for optimal placement strategies. It provides adaptive learning capabilities that effectively manage dynamic network environments and the complex, high-dimensional state space, which includes parameters like network topology, resource availability, and traffic demands.

Several advancements have refined the performance of DQNs. For instance, Double DQN [33] addressed the overestimation bias of Q-values inherent in standard DQN by decoupling action selection from Q-value evaluation, resulting in more stable and accurate learning. Similarly, Dueling DQN [34] separated the representation of the state value function and the advantage function, allowing the network to estimate better the value of specific actions in given states. However, a key challenge for most deep RL approaches is the assumption of a fully observable environment, which is often not applicable in practice. In many areas, including VNF placement, the environment is partially observable, meaning that the agent lacks complete state information at each time step. Also, temporal dependencies significantly influence network state and resource demands, leading to suboptimal decisions if not properly addressed.

To mitigate this issue, architectural or training support is essential [32,35–37]. Recent approaches incorporated Recurrent Neural Networks (RNNs) into the DQN framework to maintain a memory of past observations and actions, thus capturing critical temporal dependencies for effective VNF placement. However, RNNs can be fragile and difficult to train, often necessitating complex "warm-up" strategies to initialize hidden states at the start of each training batch [32,38].

In contrast, the Transformer model has demonstrated superior handling of temporal dependencies compared to RNNs and is increasingly prevalent [39]. Transformers are designed to handle sequential data by processing entire sequences at once rather than iteratively, as in traditional RNNs. This allows transformers to capture complex state representations and address partially observable domains. The key innovation in transformers is the self-attention mechanism, which allows the model to weigh the importance of different parts of the input sequence dynamically, enabling the model to focus on the most important parts [40]. Each element in the input sequence is transformed into three vectors: Query (Q), Key (K), and Value (V).

2025, Vol. 2, Article ID. 2025.0012
https://doi.org/10.69709/CAIC.2025.181989

These vectors are learned representations to compute attention scores. The attention score for a pair of elements is calculated as follows:

$$\text{Attention}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{\delta_k}}\right)\mathbf{V}, \tag{26}$$

where $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$ are the query, key, and value matrices, respectively, δ_k is the dimension of the keys, and $(\cdot)^T$ is the matrix transpose operator. The value vectors in matrix $\mathbf{V}$ are weighted using the weights resulting from the softmax operation. The softmax function ensures that the attention scores are normalized to sum up to one. The attention scores are then used to compute a weighted sum of the value vectors, representing the attention output and emphasizing the most relevant parts of the input sequence.

In DTQNs, the transformer network processes sequences of past states and actions to produce a rich representation of the current state. This state representation serves as an approximation of the Q-value function $Q(s,a;\theta)$, where θ denotes the network parameters. The Q-value function $Q(s,a)$ represents the expected cumulative reward for taking action a in state s and is updated iteratively using the Bellman equation as

$$\begin{aligned} &Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \\ &\alpha\left(r_t + \gamma \max_{a\in\mathcal{A}} Q(s_{t+1},a_{t+1}) - Q(s_t,a_t)\right), \end{aligned} \tag{27}$$

where α is the learning rate, r_t is the reward received at time t, γ is the discount factor, s_{t+1} is the subsequent state, and $\max_{a\in\mathcal{A}} Q(s_t,a_t)$ is the maximum Q-value for the next state over all possible actions. The network parameters are optimized by minimizing a loss function that measures the difference between predicted Q-values and target Q-values, written as

$$\begin{aligned} L(\theta) = \mathbb{E}_{(s_t,a_t,r_t,s_{t+1})\sim D}\Bigg[\Big(r_t + \gamma \max_{a\in\mathcal{A}} Q(s_{t+1},a_t;\theta^-) \\ - Q(s_t,a_t;\theta)\Big)^2\Bigg], \end{aligned} \tag{28}$$

where $\mathbb{E}$ is the expectation operator, the experience replay buffer D stores past experiences (s_t,a_t,r_t,s_{t+1}), and the target network parameters θ^- are periodically updated to stabilize training.

5.2. Proposed DTQN-Based VNF Placement Solution

This subsection presents the proposed DTQN-based VNF placement solution for incoming requests. To this end, we reformulate problem (P1) as an MDP. Specifically, we define the states, actions, and rewards as follows:

5.2.1. State

A state s_t must represent the current configuration of the network, encompassing aspects such as node capacities, resource availability, and the deployment of VNF instances. Consequently, we include the node states such as the available computing resources:

$$c_n^{\text{avail}}(t) = c_n - \sum_{k\in\mathcal{K}}\sum_{i=1}^{I^k}(\mathbf{X}_{n,k,i}(t)+\mathbf{M}_{n,k,i}(t))\cdot u^k. \tag{29}$$

We also consider the link states, such as the available bandwidth on each link

$$b_l^{\text{avail}}(t) = b_l - \sum_{r\in\mathcal{R}}\sum_{l_r\in\mathcal{L}_r}\mathbf{Y}_{l,l_r}(t)\cdot b_{l,r}. \tag{30}$$

Moreover, we consider node resource costs $\mathcal{C}_n$, node reliability α_n, link delay d_l, bandwidth cost $\mathcal{C}_l$, and link reliability α_l. Furthermore, we take into account the request states, such as the delay threshold d_r and reliability requirement α_r, and the virtual-to-physical mapping $\mathbf{Y}_{l,l_r}(t)$. Finally, we include the VNF reliability parameter α^k, the instantiation and reuse costs $\mathcal{C}_n^k$ and $\mathcal{C}_n^{i,k}$, the instantiation and reuse status matrices $\mathbf{X}_{n,k,i}(t)$ and $\mathbf{M}_{n,k,i}(t)$, and the current processing load assigned to each VNF instance on each MEC node $\mathbf{\Gamma}_{n,k,i}(t)$. Hence, the state s_t can be defined by

$$\begin{aligned} s_t = \Big(&c_n^{\text{avail}}(t),\mathcal{C}_n,\alpha_n,b_l^{\text{avail}}(t),d_l,\mathcal{C}_l,\alpha_l,\mathbf{X}_{n,k,i}(t), \\ &\mathbf{M}_{n,k,i}(t),\Gamma_{n,k,i}(t),\mathcal{C}_n^k,\mathcal{C}_n^{i,k},\alpha^k,d_r,\alpha_r,\mathbf{Y}_{l,l_r}(t)\Big). \end{aligned} \tag{31}$$

5.2.2. Actions

We define an action a_t through the following quadruple:

$$a_t = (\mathbf{X}_{n,k,i}(t),\mathbf{M}_{n,k,i}(t),\Gamma_{n,k,i}(t),\mathbf{Y}_{l,l_r}(t)). \tag{32}$$

Specifically, the DTQN agent must decide whether to instantiate a new VNF instance or reuse an existing instance, how many resources to allocate and the virtual-to-physical link mapping.

5.2.3. Reward

We implement a reward mechanism that grants positive reinforcement when the agent reduces costs compared to the previous time slot. Hence, we define this reward as

$$R_{dec} = \max(\mathcal{C}_{tot}(t-1) - \mathcal{C}_{tot}(t), 0). \tag{33}$$

Also, we penalize the agent if there are QoS violations, i.e., we define a delay violation penalty such that

$$P_{delay}(t) = \sum_{r\in\mathcal{R}}\max\left(0, \sum_{l\in\mathcal{L}}\sum_{l_r\in\mathcal{L}_r}\mathbf{Y}_{l,l_r}(t)\cdot d_l - d_r\right). \tag{34}$$

Algorithm 1 Proposed DTQN-based solution

Require: Initial state s_0, learning rate α, discount factor γ
Ensure: Optimized Q-function
1: Initialize Q-function with random weights
2: **for** $t = 1$ to T **do**
3: s'_t = *TransformerStateProcessing*(s_t) (Algo. 2)
4: Choose action a_t using ϵ-greedy strategy
5: Execute a_t and observe reward r_t and new state s_{t+1}
6: s'_{t+1} = *TransformerStateProcessing*(s_{t+1}) (Algo. 2)
7: Compute target $y_t = r_t + \gamma \max_{a_{t+1}} Q(s'_{t+1}, a_{t+1})$
8: Update Q-function:
9: $Q(s'_t, a_t) \leftarrow Q(s'_t, a_t) + \alpha(y_t - Q(s'_t, a_t))$
10: $s_t \leftarrow s_{t+1}$
11: **end for**

Algorithm 2 State processing with Transformer

Require: State s_t
Ensure: Compressed representation $\mathbf{s'_t}$
1: **function** TRANSFORMERSTATEPROCESSING(s)
2: **e** = EmbeddingLayer(s)
3: **a** = MultiHeadAttention(**e**)
4: $\mathbf{s'_t}$ = FeedForward(**a**)
5: **return** $\mathbf{s'_t}$
6: **end function**

A reliability penalty is also defined as follows:

$$P_{rel}(t) = \sum_{r\in\mathcal{R}} \max\Big(0, \alpha_r - \prod_{k\in\mathcal{K}_r}\prod_{i=1}^{I^k}(\mathbf{X}_{n,k,i}(t)\cdot\alpha^k + \mathbf{M}_{n,k,i}(t)\cdot\alpha^k) \cdot \prod_{l\in\mathcal{L}}\prod_{l_r\in\mathcal{L}_r}(\mathbf{Y}_{l,l_r}(t)\cdot\alpha_l)\Big). \quad (35)$$

Finally, the reward function $r(t)$ can now be defined as

$$r(t) = R_{dec}(t) - (\mathcal{C}_{tot}(t) + P_{delay}(t) + P_{rel}(t)). \quad (36)$$

Based on the above, we developed our DTQN-based VNF placement solution as summarized in Algorithm 1. Specifically, in Algorithm 1, we start by initializing the Q-function with random weights. At every step, the current network state is transformed into a compressed representation through the *TransformerStateProcessing* function (Lines 3 and 6 in Algorithm 1), as detailed in Algorithm 2. This function extracts relevant features to facilitate policy learning. It begins by processing the network state s_t through the embedding layer of the Transformer, converting s_t into a dense representation. The multi-head attention mechanism of the Transformer then captures complex relationships among different network entities, considering both spatial and temporal interactions. This representation is subsequently passed through the feed-forward neural network of the Transformer, resulting in the final condensed representation s'_t.

After executing the *TransformerStateProcessing* function, the remaining DTQN execution is similar to that of DQN, where the RL agent learns the Q-values by minimizing the loss function based on the Bellman equation. An action is selected based on the exploration strategy, such as the ϵ-greedy method (Line 4 in Algorithm 1). The RL agent then observes the received reward and the new state, which is also transformed into a compressed representation (Lines 5–6 in Algorithm 1). A target is calculated, comprising the observed reward and the estimated best future action, discounted by the factor γ (Line 7 in Algorithm 1). The Q-function is updated based on this target using a learning rate α (Lines 8–9 in Algorithm 1). This procedure is repeated for a specified number of steps to optimize the Q-function, ensuring a more accurate estimation of expected future rewards.

5.3. Complexity Analysis

The time complexity of the proposed DTQN algorithm arises from both the RL training loop and the Transformer-based neural network component used for Q-value approximation. In DTQN, the Transformer encoder, which processes state representations, determines the per-step computational complexity. Specifically, the self-attention mechanism within the Transformer dominates the complexity due to pairwise comparisons among input sequence elements, resulting in a quadratic dependency on the sequence length H. Combined with the embedding dimension v, the per-step complexity of the forward pass through the Transformer encoder is $O(H^2 \cdot v)$ [41]. Additional operations, such as feed-forward layers and normalization, contribute linearly or quasi-linearly to this complexity.

In the overall RL training loop, each step involves sampling a mini-batch of size G from the replay buffer, executing forward passes through the Transformer-based model for both the current and next states, and performing a backward pass to update the network parameters. The forward and backward passes share a similar computational cost, making the per-step training complexity approximately $O(G \cdot H^2 \cdot v)$. Over T_{train} total training steps, the overall complexity is $O(T_{train} \cdot G \cdot H^2 \cdot v)$.

Hence, the computational intensity of the DTQN arises primarily from the sequence modeling capabilities

2025, Vol. 2, Article ID. 2025.0012
https://doi.org/10.69709/CAIC.2025.181989

of the Transformer encoder. While this architecture entails higher computational costs than traditional methods, its ability to effectively capture complex temporal and spatial dependencies greatly enhances the accuracy and efficiency of decision-making in VNF placement and redeployment.

6. Experimental Results and Discussion

This section presents the simulations conducted to evaluate the performance of our proposed solution against existing baselines. We first describe the simulation environment, followed by an analysis of the obtained results.

6.1. Simulation Setup

We conduct our simulations on Google Colab using the Tesla T4 GPU, which features a Turing architecture with 2560 CUDA cores and 320 Tensor cores. This GPU provides up to 8.1 Floating Point Operations Per Second (TFLOPS) of single precision floating point performance, combined with 16 GB of GDDR6 memory and a bandwidth of up to 320 GB/s.

The simulation parameters, including network configurations, request characteristics, and hyperparameter values, were selected based on representative benchmarks and practices in MEC and NFV research, as well as reinforcement learning standards, such as [42–44], to ensure realistic and challenging test conditions.

Our substrate network comprises users uniformly distributed across the network and [5–10] MEC nodes, with each MEC server featuring one to five outgoing links, randomly assigned to other MEC nodes to reflect network topology variability and dynamism. Each MEC node is configured as a multi-core system with a maximum computing resource capacity ranging from 5 to 10 GHz across all cores, a unit resource cost of 10, and a reliability parameter ranging from 99.9% to 99.999%. Note that in our simulations, we intentionally limit the number of MECs and deliberately set the resource capacity of nodes to provoke migration scenarios.

Each physical communication link is assigned a bandwidth capacity ranging from 100 to 200 Mbps, with a bandwidth cost of 0.05 per Mb and a reliability rate between 95% and 99.9%. The required processing capacity for the VNFs ranges from 1 to 4 GHz, and the resource storage capacity for each VNF ranges from 50 to 200 MB. The VNF reliability requirement is set between 99.9% and 99.999%, reflecting the stringent requirements of URLLC applications. The instantiation cost for a Virtual Network Function (VNF) is fixed at 10, while the reuse cost is set at 5.

In our experiments, we consider incoming service requests from end-users that require the deployment of a VNF-based Service Function Chain (SFC) in the Multi-access Edge Computing (MEC) network. These service requests are categorized into four types, depending on the number of VNFs within the service chain, ranging from 1 to 4 VNFs. Service requests are generated following a Poisson distribution with an arrival rate parameter δ that varies within the set {30, 50, 80, 100}. Each request has a delay threshold between 1 and 10 milliseconds (ms), and the reliability requirement is randomly selected from the range [99%, 99.9%].

The proposed solution, referred to as DTQN, is evaluated against three baseline algorithms: a DQN-based method (DQN), a heuristic approach relying on random placement (Heuristic), and a D3T-based method [30], which has been modified to incorporate redeployment strategies to ensure fairness. The DTQN architecture has an input layer sized to the state vector and an embedding layer with a size of 128. It is followed by a Transformer encoder layer, which includes a multi-head attention mechanism with two heads, each having a size of 64. The feed-forward network within each transformer encoder layer consists of two fully connected layers, each containing 128 ReLU-activated neurons. Layer normalization and dropout (with a rate of 0.1) are applied after each sub-layer in the transformer encoder. The final output layer is a fully connected layer that generates Q-values with a size corresponding to the number of potential actions. The D3T architecture builds on the Transformer-based approach, similar to DTQN, but differs in its decision-making process. It includes a single transformer encoder layer with two multi-head attention heads and a feed-forward network of 128 neurons per layer, ultimately outputting Q-values for action selection. The DQN architecture is configured with an input layer sized to the state vector and includes three fully connected layers, each with 128 ReLU neurons. The final output layer is a fully connected layer that produces Q-values, sized to the number of actions. The used hyperparameters for the DTQN, D3T and DQN are summarized in the Table 1. To incentivize reuse over migration, we assign a reward factor of 7 for reuse and 4 for migration. Conversely, failing to fulfill a request due to a violation of QoS requirements incurs a penalty with a reward factor of -5.

6.2. Simulation Results and Discussion

We begin by examining the impact of increasing the number of nodes and the arrival rates δ for each algorithm. The

2025, Vol. 2, Article ID. 2025.0012
https://doi.org/10.69709/CAIC.2025.181989

Table 1: Selected hyperparameters for learning-based methods

Hyperparameter	Value
Replay memory size	20000
Discount factor	0.99
Exploration rate	1.0, decaying to 0.01
Exploration rate decay	0.995
Learning rate	0.0005
Batch size	256
Number of episodes	1000
Frequency of target update	500

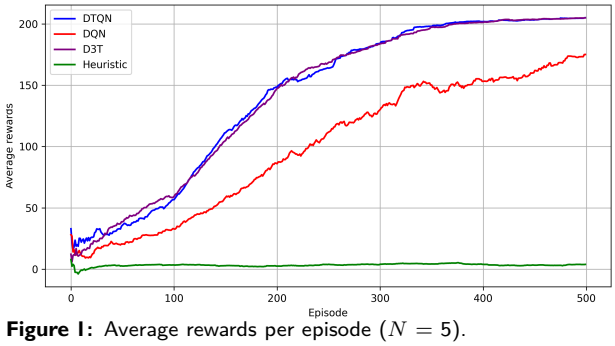

Figure 1: Average rewards per episode ($N = 5$).

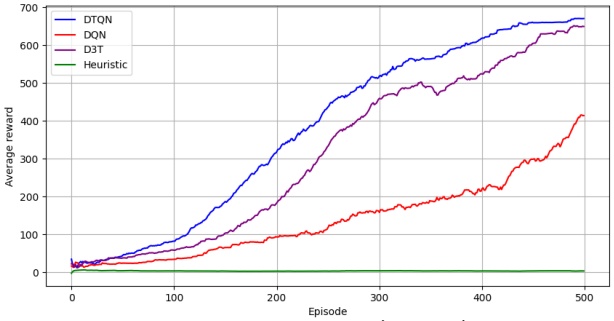

Figure 2: Average rewards per episode ($N = 10$).

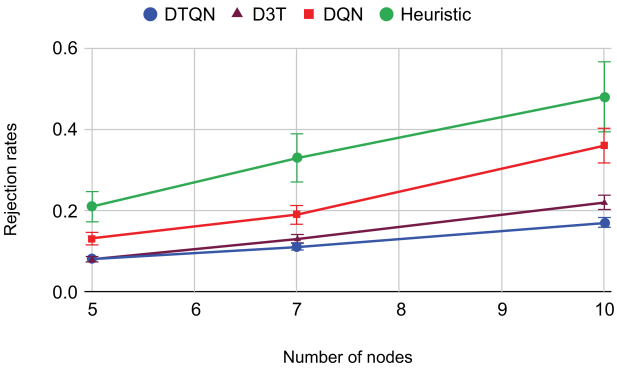

Figure 3: Rejection rates vs number of nodes ($\delta = 50$).

average rewards per episode for each algorithm are presented in Figure 1, which considers 5 nodes with $\delta = 30$, while in Figure 2, we consider 10 nodes with $\delta = 100$. It is important to note that in each episode, network link mappings among nodes vary to capture the variability and dynamism of the network topology. Furthermore, strict adherence to QoS requirements is enforced for all requests, and partial satisfaction is not accepted. Consequently, only fully accepted requests are included in the results presented in Figures 1 and 2.

For any δ (Figures 1 and 2), the proposed DTQN solution consistently achieves the highest rewards, despite its greater computational demands and longer training times, compared to baseline algorithms such as DQN and Heuristic. While DTQN and D3T exhibit similar reward performance at a smaller scale, DTQN surpasses D3T as the number of users increases. In contrast, the Heuristic approach consistently yields the lowest rewards. This trend is further supported by Figure 3, which presents the average rejection rates over five simulation runs with $\delta = 50$. The results indicate that DTQN achieves the lowest rejection rate, followed by D3T and DQN, while the Heuristic approach has the highest rejection rate.

Notably, the performance gap between DTQN, D3T, and DQN in Figures 1 and 2 widen as the number of nodes in the topology and δ increase. This is likely due to the enhanced ability of DTQN to capture temporal dependencies, eventually leading to more accurate SFC placement decisions that prioritize reuse over migration. As the request arrival rate δ increases and network resources become saturated, optimizing VNF placement and reuse decisions becomes crucial to avoid network overloading.

This observation is further validated by Figure 4 ($\mathcal{N} = 10$, $\delta = 30$) and Figure 5 ($\mathcal{N} = 10$, $\delta = 100$), which illustrate the SFC embedding strategies adopted by each evaluated approach. The Heuristic method, which achieves the lowest average rewards, exhibits a uniform distribution across new VNF instantiation, migration, and reuse placement decisions, regardless of the arrival rate δ. In contrast, the DQN baseline demonstrates a more adaptive strategy, adjusting the balance between VNF placement and reuse based on the value of δ. Meanwhile, DTQN and D3T-based solutions consistently prioritize VNF instance reuse across all values of δ. As the number of users increases, DTQN further distinguishes itself from D3T by prioritizing VNF instance reuse even more.

This preference can be attributed to several factors. First, the simulation parameters set the reuse cost lower

2025, Vol. 2, Article ID. 2025.0012
https://doi.org/10.69709/CAIC.2025.181989

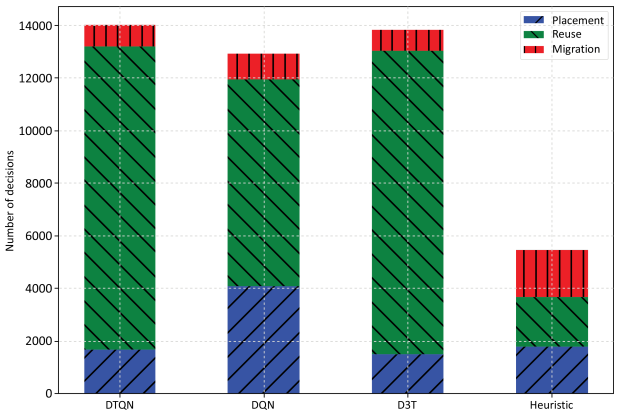

Figure 4: Distribution of SFC embedding decisions ($N = 10$ and $\delta = 30$).

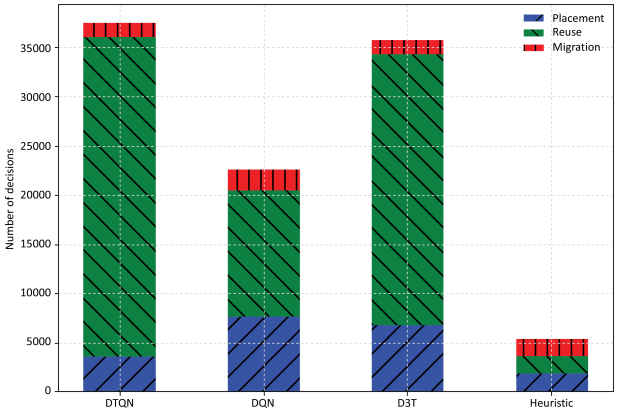

Figure 5: Distribution of SFC embedding decisions ($N = 10$ and $\delta = 100$).

than the migration cost, making VNF reuse more cost-effective than instantiating new VNFs. Moreover, migration induces extra bandwidth consumption, potentially jeopardizing QoS requirements and incurring penalties, particularly at high δ. By prioritizing VNF reuse, the DTQN-based solution minimizes unnecessary resource allocations and bandwidth usage, thereby maintaining a more stable and efficient MEC network. Furthermore, the enhanced ability of our solution to capture temporal dependencies enables more informed placement decisions, striking a strategic balance between immediate resource utilization and long-term network performance.

Figure 6, which illustrates the average costs of each algorithm as a function of the number of nodes over five simulation runs, supports the previous statement ($\delta = 50$). Despite having the highest rejection rate compared to DTQN, D3T and DQN, the Heuristic approach incurs the highest costs due to its inadequate placement strategy and excessive bandwidth consumption. At smaller scales,

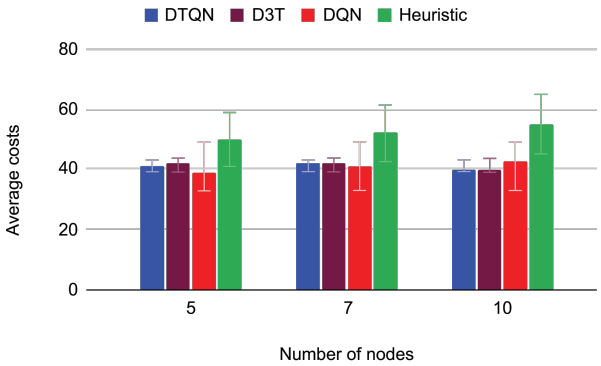

Figure 6: Average costs vs number of nodes ($\delta = 50$).

DQN incurs slightly lower costs than DTQN. However, it is imperative to note that DQN has a higher rejection rate, resulting in fewer satisfied requests. Moreover, as the number of nodes increases, DTQN's costs decrease relative to DQN's, since DQN's suboptimal decisions have a greater impact than the difference in the number of requests served.

These findings suggest that Transformer-based RL models, such as DTQN and D3T, hold significant promise for complex decision-making tasks in dynamic networks, particularly in optimizing VNF placement strategies. By prioritizing VNF reuse over new instantiation and migration, DTQN achieves superior performance, cost efficiency, and resource optimization, especially under varying network demands.

7. Conclusions

This paper presents a comprehensive solution for VNF placement and utilization in MEC networks, specifically tailored for mission-critical applications with stringent QoS requirements. Our DTQN-based approach effectively addressed the dynamic nature of MEC environments by leveraging advanced RL techniques to optimize VNF placement strategies. Simulation results demonstrated the superiority of the proposed method in achieving cost efficiency, minimizing service disruptions, and ensuring better resource utilization compared to baseline approaches. The emphasis on reusing VNFs rather than creating new instances or migrating VNFs highlights the crucial role of smart resource management in supporting dynamic networks and minimizing operational expenses. These findings suggest that Transformer-based RL models are promising for solving complex decision-making tasks in dynamic and resource-constrained network environments. In future work, we will investigate the scalability challenge of the proposed DTQN-based solution in large-scale network environments.

Abbreviations

5G	5th Generation
APMEC	Automated Provisioning Framework for MEC
DT	Decision Transformer
DDQN	Double Deep Q-Network
DQN	Deep Q-Network
GAP	Generalized Assignment Problem
IoT	Internet-of-Things
IT	Information Technology
MDP	Markov Decision Process
MEC	Multi-Access Edge Computing
MILP	Mixed-Integer Linear Programming
MINLP	Mixed-Integer Non-Linear Programming
ML	Machine Learning
NFV	Network Function Virtualization
QoS	Quality-of-Service
RNN	Recurrent Neural Network
RL	Reinforcement Learning
SAP	Subchain-Aware NFV Service Placement
SFC	Service Function Chain
SLA	Service-Level Agreement
TFLOPS	Floating Point Operations Per Second
URLLC	Ultra-Reliable Low-Latency Communication
VM	Virtual Machine
VNF	Virtual Network Function

Author Contributions

Conceptualization and supervision, W.J. and F.B.; methodology, A.N. and M.D.; validation, R.G. and F.B.; formal analysis and investigation, A.N. and M.D.; resources, W.J.; writing—original draft preparation, A.N. and M.D.; writing—review and editing, W.J., F.B. and R.G.; project administration, W.J. and F.B.; funding acquisition, W.J. and F.B. All authors have read and agreed to the published version of the manuscript.

Availability of Data and Materials

Data and materials supporting the results of this study are available upon request from the corresponding author.

Conflicts of Interest

The authors declare no conflicts of interest regarding this manuscript.

Funding

This work was funded by the Study in Canada Scholarship program.

Acknowledgments

Declared None.

References

[1] N. Alliance, "*Verticals URLLC Use Cases and Requirements*," NGMN Alliance, Düsseldorf, Germany, 2019. Available online: https://www.ngmn.org/publications/verticals-urllc-use-cases-and-requirements.html.

[2] A. A. Barakabitze, A. Ahmad, R. Mijumbi, and A. Hines, "5G network slicing using SDN and NFV: A survey of taxonomy, architectures and future challenges," *Elsevier Comput. Netw.*, vol. 167, p. 106984, 2020. [CrossRef]

[3] W. Z. Khan, E. Ahmed, S. Hakak, I. Yaqoob, and A. Ahmed, "Edge computing: A survey," *Future Gener. Comput. Syst.*, vol. 97, pp. 219–235, 2019. [CrossRef]

[4] H. Guo, Y. Wang, Z. Li, X. Qiu, H. An, P. Yu, and N. Yuan, "Cost-aware placement and chaining of service function chain with VNF instance sharing," in *Proceedings IEEE/IFIP Network Operations and Management Symposium (NOMS)*, Budapest, Hungary, 2020. [CrossRef]

[5] A. Mseddi, W. Jaafar, H. Elbiaze, and W. Ajib, "Joint container placement and task provisioning in dynamic fog computing," *IEEE IoT J.*, vol. 6, no. 6, pp. 10028–10040, 2019. [CrossRef]

[6] A. Mseddi, W. Jaafar, H. Elbiaze, and W. Ajib, "Intelligent resource allocation in dynamic fog computing environments," in *Proceedings IEEE International Conference on Cloud Networking (CloudNet)*, Coimbra, Portugal, 2019, pp. 1–7. [CrossRef]

[7] M. Dieye, W. Jaafar, H. Elbiaze, and R. H. Glitho, "Market driven multidomain network service orchestration in 5G networks," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 7, pp. 1417–1431, 2020. [CrossRef]

[8] M. Dieye, A. Mseddi, W. Jaafar, and H. Elbiaze, "Towards reliable remote health monitoring in fog computing networks," *IEEE Trans. Netw. Serv. Mngt.*, vol. 19, no. 3, pp. 2506–2520, 2022. [CrossRef]

[9] M. Dieye, W. Jaafar, H. Elbiaze, and R. H. Glitho, "DRL-based green resource provisioning for 5G and beyond networks," *IEEE Trans. Green Commun. Netw.*, vol. 7, no. 4, pp. 2163–2180, 2023. [CrossRef]

[10] A. Mseddi, W. Jaafar, H. Elbiaze, and W. Ajib, "Centralized and collaborative rl-based resource allocation in virtualized dynamic fog computing," *IEEE IoT J.*, vol. 10, no. 16, pp. 14239–14253, 2023. [CrossRef]

[11] B. Li, B. Cheng, X. Liu, M. Wang, Y. Yue, and J. Chen, "Joint resource optimization and delay-aware virtual network function migration in data center networks," *IEEE Trans. Netw. Serv. Mngt.*, vol. 18, no. 3, pp. 2960–2974, 2021. [CrossRef]

[12] S. N. Afrasiabi, A. Ebrahimzadeh, N. Promwongsa, C. Mouradian, W. Li, A. Recse, R. Szabo, and R. H. Glitho, "Cost-efficient cluster migration of VNFs for service function chain embedding," *IEEE Trans.*

Netw. Serv. Mngt., vol. 21, no. 1, pp. 979–993, 2024. [CrossRef]

[13] S. Zhang, W. Jia, Z. Tang, J. Lou, and W. Zhao, "Efficient instance reuse approach for service function chain placement in mobile edge computing," *Elsevier Comput. Netw.*, vol. 211, p. 109010, 2022. [CrossRef]

[14] N. Kiran, X. Liu, S. Wang, and C. Yin, "VNF placement and resource allocation in SDN/NFV-enabled MEC networks," in *Proceedings IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, Seoul, Republic of Korea, 2020, pp. 1–6. [CrossRef]

[15] D. Li, P. Hong, K. Xue, and J. Pei, "Virtual network function placement and resource optimization in NFV and edge computing enabled networks," *Elsevier Comput. Netw.*, vol. 152, pp. 12–24, 2019. [CrossRef]

[16] D. Chemodanov, P. Calyam, and F. Esposito, "A near-optimal reliable composition approach for geo-distributed latency-sensitive service chains," in *Proceedings IEEE Conference on Computer Communications (INFOCOM)*, Paris, France, 2019, pp. 1792–1800. [CrossRef]

[17] M. Dieye, S. Ahvar, J. Sahoo, E. Ahvar, R. Glitho, H. Elbiaze, and N. Crespi, "CPVNF: Cost-efficient proactive VNF placement and chaining for value-added services in content delivery networks," *IEEE Trans. Netw. Serv. Mngt.*, vol. 15, no. 2, pp. 774–786, 2018. [CrossRef]

[18] A. Filali, A. Abouaomar, S. Cherkaoui, A. Kobbane, and M. Guizani, "Multi-access edge computing: A survey," *IEEE Access*, vol. 8, pp. 197017–197046, 2020. [CrossRef]

[19] H. Djigal, J. Xu, L. Liu, and Y. Zhang, "Machine and deep learning for resource allocation in multi-access edge computing: A survey," *IEEE Commun. Surv. Tutor.*, vol. 24, no. 4, pp. 2449–2494, 2022. [CrossRef]

[20] P. Cruz, N. Achir, and A. C. Viana, "On the edge of the deployment: A survey on multi-access edge computing," *ACM Comput. Surv.*, vol. 55, no. 5, pp. 1–34, 2022. [CrossRef]

[21] F. Spinelli and V. Mancuso, "Toward enabled industrial verticals in 5G: A survey on MEC-based approaches to provisioning and flexibility," *IEEE Commun. Surv. Tutor.*, vol. 23, no. 1, pp. 596–630, 2020. [CrossRef]

[22] L. Yala, P. A. Frangoudis, and A. Ksentini, "Latency and availability driven VNF placement in a MEC-NFV environment," in *Proceedings IEEE Global Communications Conference (GLOBECOM)*, Abu Dhabi, United Arab Emirates, 2018, pp. 1–7. [CrossRef]

[23] P. Jin, X. Fei, Q. Zhang, F. Liu, and B. Li, "Latency-aware VNF chain deployment with efficient resource reuse at network edge," in *Proceedings IEEE Conference on Computer Communications (INFOCOM)*, Toronto, ON, Canada, 2020, pp. 267–276. [CrossRef]

[24] D. Cho, J. Taheri, A. Y. Zomaya, and P. Bouvry, "Real-time virtual network function (VNF) migration toward low network latency in cloud environments," in *Proceedings IEEE International Conference on Cloud Computing (CLOUD)*, Honolulu, , HI, USA, 2017, pp. 798–801. [CrossRef]

[25] T.-W. Kuo, B.-H. Liou, K. C.-J. Lin, and M.-J. Tsai, "Deploying chains of virtual network functions: On the relation between link and server usage," *IEEE/ACM Trans. Netw.*, vol. 26, no. 4, pp. 1562–1576, 2018. [CrossRef]

[26] H. Zhu, V. Gupta, S. S. Ahuja, Y. Tian, Y. Zhang, and X. Jin, "Network planning with deep reinforcement learning," in *Proceedings ACM SIGCOMM Conference*, New York, NY, USA, 2021, pp. 258–271. [CrossRef]

[27] T. V. Doan, G. T. Nguyen, M. Reisslein, and F. H. Fitzek, "SAP: Subchain-aware NFV service placement in mobile edge cloud," *IEEE Trans. Netw. Serv. Mngt.*, vol. 20, no. 1, pp. 319–341, 2022. [CrossRef]

[28] A. Abouaomar, Z. Mlika, A. Filali, S. Cherkaoui, and A. Kobbane, "A deep reinforcement learning approach for service migration in mec-enabled vehicular networks," in *Proceedings IEEE Conference on Local Computer Networks (LCN)*, Edmonton, AB, Canada, 2021, pp. 273–280. [CrossRef]

[29] T. Subramanya, D. Harutyunyan, and R. Riggio, "Machine learning-driven service function chain placement and scaling in MEC-enabled 5G networks," *Elsevier Comput. Netw.*, vol. 166, p. 106980, 2020. [CrossRef]

[30] B. Wu, D. Chen, N. V. Abhishek, and M. Gurusamy, "D3T: Double deep Q-network decision transformer for service function chain placement," in *Proceedings IEEE International Conference on High-Performance Switching and Routing (HPSR)*, Albuquerque, NM, USA, 2023, pp. 167–172. [CrossRef]

[31] R. Cohen, L. Katzir, and D. Raz, "An efficient approximation for the generalized assignment problem," *Info. Process. Lett.*, vol. 100, no. 4, pp. 162–166, 2006. [CrossRef]

[32] K. Esslinger, R. Platt, and C. Amato, "Deep transformer Q-networks for partially observable reinforcement learning," *arXiv*, arXiv:2206.01078, 2022. [CrossRef]

[33] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proceedings AAAI Conference on Artificial Intelligence*, Phoenix, AZ, USA, vol. 30, no. 1, 2016. [CrossRef]

[34] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas, "Dueling network architectures for deep reinforcement learning," in *Proceedings International Conference on Machine Learning (ICML)*, New York City, NY, USA, 2016, pp. 1995–2003. [CrossRef]

[35] L. Pinto, M. Andrychowicz, P. Welinder, W. Zaremba, and P. Abbeel, "Asymmetric actor critic for image-based robot learning," in *Robotics: Science and Systems XIV*, Pittsburgh, PA, USA, 2018. H. Kress-Gazit, S. S. Srinivasa, T. Howard and N. Atanasov, Eds. [CrossRef]

[36] M. Igl, L. Zintgraf, T. A. Le, F. Wood, and S. Whiteson, "Deep variational reinforcement learning for POMDPs," in *Proceedings International Conference on Machine Learning (ICML)*, Stockholm, Sweden, 2018, pp. 2117–2126. [CrossRef]

[37] X. Ma, P. Karkus, D. Hsu, W. S. Lee, and N. Ye, "Discriminative particle filter reinforcement learning for complex partial observations," in *Proceedings International Conference on Learning Representations (ICLR)*, Addis Ababa, Ethiopia, 2020. [CrossRef]

[38] G. Lample and D. S. Chaplot, "Playing fps games with deep reinforcement learning," *Proceedings AAAI Conference on Artificial Intelligence*, San Francisco, CA, USA, vol. 31, no. 1, 2017. [CrossRef]

[39] U. Upadhyay, N. Shah, S. Ravikanti, and M. Medhe, "Transformer based reinforcement learning for games," *arXiv*, arXiv:1912.03918 2019. [CrossRef]

[40] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *Proceedings International Conference on Learning Representations (ICLR)*, San Diego, CA, USA, 2015. Y. Bengio and Y. LeCun, Eds. [CrossRef]

[41] S. Wang, B. Z. Li, M. Khabsa, H. Fang, and H. Ma, "Linformer: Self-attention with linear complexity," *arXiv*, arXiv:2006.04768, 2020. [CrossRef]

[42] M. E. Haque, F. Tariq, M. R. Khandaker, K.-K. Wong, and Y. Zhang, "A survey of scheduling in 5G URLLC and outlook for emerging 6g systems," *IEEE Access*, vol. 11, pp. 34372–34396, 2023. [CrossRef]

[43] T. Eimer, M. Lindauer, and R. Raileanu, "Hyperparameters in reinforcement learning and how to tune them," in *International Conference on Machine Learning—PMLR*, Honolulu, HI, USA, 2023, pp. 9104–9149. [CrossRef]

[44] R. Sahraoui, O. Houidi, and F. Bannour, "Energy-aware vnf-fg placement with transformer-based deep reinforcement learning," in *2024 IEEE/IFIP Network Operations and Management Symposium (NOMS 2024)*, Seoul, Republic of Korea, 2024. [CrossRef]