

MARC-6G: Multi-Agent Reinforcement Learning for Distributed Context-Aware SFC Deployment and Migration in 6G Networks

Solomon Fikadie Wassie, Eric Samikwa, Antonio Di Maio, and Torsten Braun

Institute of Computer Science, University of Bern, Switzerland

Email: {solomon.wassie, eric.samikwa, antonio.dimaio, torsten.braun}@unibe.ch

Abstract—The Cloud Continuum Framework (CCF) extends computing capabilities across near-edge, far-edge, and extreme-edge nodes beyond the traditional edge to meet the diverse performance demands of emerging 6G applications. While Deep Reinforcement Learning (DRL) has demonstrated potential in automating Virtual Network Function (VNF) migration by learning optimal policies, centralized DRL-based orchestration faces challenges related to scalability and limited visibility in distributed, heterogeneous network environments. To address these limitations, we introduce MARC-6G (Multi-Agent Reinforcement Learning for Distributed Context-Aware Service Function Chain (SFC) Deployment and Migration in 6G Networks), a novel framework that leverages decentralized agents for distributed, dynamic, and service-aware SFC placement and migration. MARC-6G allows agents to monitor different portions of the network, collaboratively optimize network control policies via experience sharing, and make local decisions that collectively enhance global orchestration under time-varying traffic conditions. We show through simulations that MARC-6G improves SFC deployment efficiency, reduces migration costs by 34%, and lowers energy consumption by 12.5% compared to the state-of-the-art centralized DRL baseline.

Index Terms—Multi-Agent Reinforcement Learning, Distributed Service Orchestration, Distributed Intelligence, Service Function Chain

I. INTRODUCTION

The Sixth Generation (6G) mobile communication network is expected to leverage the concept of the Cloud Continuum Framework (CCF), which provides more flexible computational resources closer to end users beyond traditional edge computing, thereby meeting diverse application requirements [1]. However, realizing the full potential of this continuum requires scalable and intelligent orchestration of distributed resources across a heterogeneous, multi-tier infrastructure.

Network services are provisioned as sequences of heterogeneous, predefined, and ordered Virtual Network Function (VNF) in the form of SFC on standardized, general-purpose servers enabled by Network Function Virtualization (NFV) technology [2]. VNFs are software-based implementations of network services such as Network Address Translation (NAT), Firewalls (FW), Intrusion Detection and Prevention Systems (IDPS), WAN optimizers (WO), Video Optimization Controllers (VOC), Traffic monitors (TM), and encoding/decoding functionalities. Those network functions provide a wide range of emerging applications, including video streaming, virtual/augmented/mixed

reality, Industry 4.0, holographic communication, smart factories, autonomous vehicles, tactile industrial internet [3].

Optimal VNF deployment is one of the design requirements of modern mobile networks to ensure sustainable long-term performance and minimize operational costs. This, in turn, enables fast, reliable, and cost-effective delivery of network services. Several studies have proposed machine learning based approaches for the centralized orchestration of network functions [4], [5]. Deep Reinforcement Learning (DRL) is employed for network state awareness by leveraging Deep Learning (DL) to extract complex, high-dimensional network patterns and using Reinforcement Learning (RL) to optimize decision-making through interactions with dynamic network states.

A Service Orchestrator (SO) is a network management system designed to automate the provisioning, scaling, and lifecycle management of network services [6]. Despite being effective in small-scale networks, centralized service orchestrators exhibit several limitations in large-scale environments due to their limited visibility of the global network state. These limitations include a single point of failure, high signaling overhead for network-wide data collection, and reduced responsiveness in real-time decision-making, which degrade the performance of latency-sensitive applications, and often lack the flexibility and scalability required to efficiently manage dynamic and heterogeneous workload demands [7].

The main research question addressed in this work is: **How to optimally deploy and dynamically reconfigure multiple SFC requests in large-scale 6G networks, while adapting to time-varying traffic demands and heterogeneous infrastructure resources, to satisfy end-to-end performance requirements?**

To address this challenge, we introduce MARC-6G (Multi-Agent Reinforcement Learning for Distributed Context-Aware SFC Deployment and Migration in 6G Networks). In MARC-6G, agents monitor portions of the CCF and collaboratively learn VNF placement and migration policies from real-time network metrics, enabling scalable and dynamic orchestration across heterogeneous 6G infrastructures. The key contributions of this paper are summarized as follows:

- We model the problem of scalable and distributed service-aware orchestration of VNF deployment and migration for multiple SFC requests in large-scale 6G networks.
- We design multi-agent RL-based distributed orchestrators

that use local state to jointly learn deployment and migration policies, minimizing delay, energy consumption, and VNF migration cost under dynamic traffic and resource conditions, while concurrently provisioning multiple SFCs. provisioning multiple SFCs.

- We evaluate MARC-6G against baseline methods and demonstrate higher request acceptance, improved energy efficiency, and reduced migration costs, validating its effectiveness for dynamic SFC management in 6G networks.

The remainder of the paper is organized as follows: Section II describes the related works. Section III presents the system model and problem formulation. Section IV outlines the proposed methods. Section V presents the performance evaluation. Finally, Section VI draws the conclusions.

II. RELATED WORKS

Existing adaptive centralized provisioning techniques address VNF deployment as an elastic resource provisioning problem, aiming for flexible and on-demand resource allocation to meet network service requirements and service level agreements [4], [8], [9]. However, they often overlook the fact that multiple service requests may arrive at the orchestrator simultaneously, with varying performance requirements.

Tang et al. [10] employ a digital twin powered by an attention model to guide an RL agent by predicting resource requirements a priori. However, because prediction is decoupled from action selection, the system's ability to adapt in real time is compromised. Onsu et al. [8] apply DRL for VNF placement using fixed data center priorities based on residual capacity, but static scoring overlooks context and traffic, resulting in suboptimal placement.

Dynamic priority assignment enables more adaptive and scalable orchestration under resource variability. Tanuboddi et al. [11] addressed VNF migration by leveraging software-based network functions to enable dynamic scaling, facilitating seamless migration in response to user mobility, load variations, and hardware failures. Chen et al. [12] and J.Chen et al. [13] address cost-efficient and fault-tolerant SFC migration using DRL and optimization techniques, respectively, but both approaches overlook key aspects such as fairness, realistic service lifetimes. Table I provides a comparison of the parameters considered in this study with those reported in the literature.

III. SYSTEM MODEL AND PROBLEM FORMULATION

A. Distributed Service Orchestration in 6G Networks

We envision the 6G cloud network architecture that consists of three frameworks: the CCF, the Management and Orchestration Framework (MOF), and the Artificial Intelligence and Machine Learning Framework (AIMLF), as shown in Fig. 1 [15]. The CCF provides logically unified resource management across cloud-to-edge environments by dynamically integrating resources into Cloud, Near-edge, Far-edge, and Extreme-edge. A portion of CCF is highlighted in a different color to illustrate that VNFs of a single network service can be deployed across heterogeneous CCF nodes.

TABLE I: Comparison of Related Works.

Reference	Concurrent VNF Migration	Multiple SFC Requests	Stateful VNF Migration	Migration Cost
Tang et al. [10]	✓	×	✓	×
J.Chen et al. [13]	✓	×	✓	✓
Onsu et al. [8]	×	✓	×	×
Tanub. et al. [11]	×	✓	×	✓
Zhang et al. [14]	×	✓	✓	×
S.Long et al. [4]	×	✓	×	×
Chen et al. [12]	✓	×	✓	✓
Liu et al. [9]	✓	✓	×	×
MARC-6G	✓	✓	✓	✓

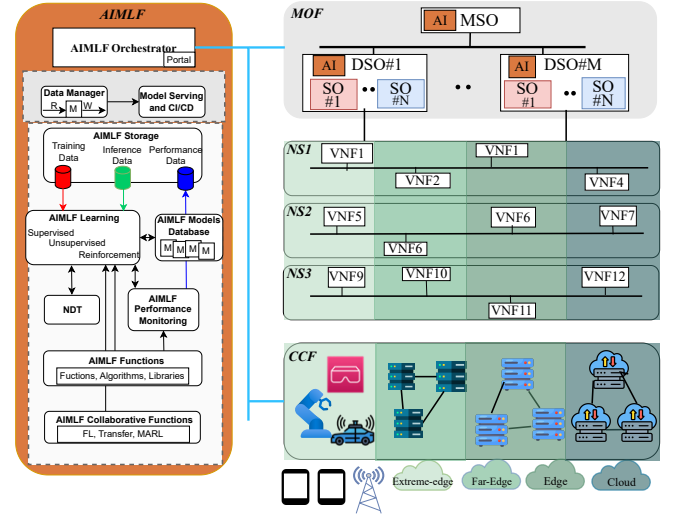


Fig. 1: AI-native 6G network architecture with distributed service orchestrators for scalable network state adaptive VNF deployment [15].

The MOF provides distributed orchestration capabilities to enable scalable orchestration, interfacing with the CCF for infrastructure control and with the AIMLF for learning-based decision support. It comprises two components: the Master Service Orchestrator (MSO) and the Distributed Service Orchestrators (DSOs). The MSO is responsible for the initial deployment of Network Services (NSs) and DSOs across the CCF, while the DSOs manage runtime operations and the network service lifecycle management. Each DSO contains multiple SO to address the dynamic workloads and scalability challenges posed by the heterogeneous 6G network infrastructure.

The AIMLF is the intelligent control framework, supporting real-time monitoring and continuous learning in dynamic network environments. It ensures autonomous orchestration and adaptive service management by cooperating with the CCF and MOF. DSOs comprise intelligent agents that leverage the AIMLF to manage the CCF in real time, enabling context-aware decisions. DSOs monitor resource availability to support autonomous VNF allocation, predictive scaling, and proactive SFC migration. This ensures low latency, energy efficiency, and enhanced resilience and fault tolerance.

B. System Model

We model distributed orchestration over the CCF at a high level (Fig. 2), where service orchestrators manage batches of SFC requests in a queue. We consider CCF network infrastructure that comprises heterogeneous physical nodes v_i , each with CPU/GPU capacity C_i [cycles/s], distributed across four tiers: (i) centralized cloud data centers, (ii) near-edge nodes, (iii) far-edge nodes, and (iv) end-user devices.

We represent the network infrastructure as a weighted undirected graph $G = (V, E, W)$, where V is the set of physical nodes, $E \subseteq V \times V$ denotes the set of links connecting them, and the *weight function* $W : E \rightarrow \mathbb{R}^+$ represents each link's available bandwidth. Each node $v \in V$ represents a physical network entity, such as an extreme-edge device (e.g., smartphone, electric vehicle, or drone), an edge or near-edge server, or a centralized cloud data center within the CCF. Each link $e \in E$ represents a high-speed communication path, typically implemented via fiber connections. We denote the bandwidth capacity between nodes $v_i, v_j \in V$ as B_{ij} [bit/s].

We consider a scenario involving multiple SFC requests arriving at the orchestrator. Each SFC request is modeled as a Directed Acyclic Graph (DAG) $f_i = (K_i, L_i, \delta_i, \zeta_i, \Lambda_i, B_i^{\min}, D_i^{\max}, \sigma_i)$, where K_i denotes the set of VNFs for the i -th SFC; L_i denotes the set of logical links between VNFs; δ_i and ζ_i denote the source and destination endpoints, respectively; Λ_i signifies the traffic arrival time; $B_i^{\min}$ [bit/s] is the minimum bandwidth requirement; $D_i^{\max}$ [s] is the maximum tolerable end-to-end delay; and σ_i [cycle/s] is the total computational demand across all VNFs. Each VNF $k \in K_i$ represents a softwarized network function that can process incoming packets. The logical links $(k_i, k_j) \in L_i$ represent the connections between successive VNFs k_i and k_j , which represent a sequential dependency between VNFs.

To support concurrent deployment, we consider a batch of N active SFC requests simultaneously for deployment, denoted by $F = (f_1, f_2, \dots, f_N)$. These requests, predefined according to application-specific requirements and submitted by tenants, are orchestrated in parallel over the physical infrastructure. The topology f_i of an SFC is determined by the application it serves and is assumed to be specified by the tenant and forwarded to the network management plane for processing and deployment.

Each VNF $k \in K_i$ maintains an internal state, making *stateful migration* essential to preserve session continuity and avoid service disruption during reallocation. Given user mobility and fluctuating link quality, proactive resource management and adaptive state transfer are essential. The selection of a target node for migrating a stateful VNF can be modeled as a tuple $\mathcal{S}_k = (M_i, D_c, Q_v, P_s, T_m)$ where M_i is the size of the context to be migrated, D_c represents the deployment cost of the service on the target node, Q_v denotes the SLA violation impact during migration, P_s indicates the congestion level along the selected migration path, and T_m is the total migration time.

C. Problem Formulation

We formulate the problem of simultaneous, elastic self-scaling placement of VNFs for a batch of N active SFC requests

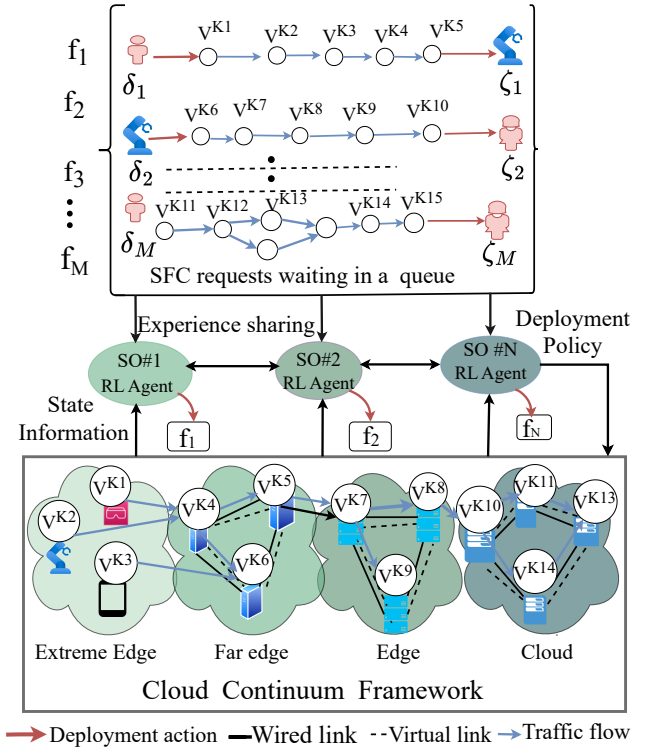


Fig. 2: Several SFC requests arrive at the orchestrator in queues, while multiple intelligent agents concurrently deploy VNFs over CCF.

over a shared physical network, with the goal of determining the *optimal placement* that minimizes end-to-end SFC latency. The total end-to-end delay of an SFC typically comprises propagation, communication, queuing, VNF computing, and virtualization delays. For tractability, we simplify our formulation by considering communication delay, VNF computing delay, and queuing delay. Our objective is to determine an *optimal allocation* that minimizes overall system latency, energy consumption, and migration cost.

For each SFC request f_i , we define the *allocation vector* as $\alpha_i = (\alpha_{i,1}, \alpha_{i,2}, \dots, \alpha_{i,|K_i|}) \in V^{|K_i|}$, where each element $\alpha_{i,j} \in V$ represents the physical node on which the j -th VNF for the SFC request f_i is deployed. The complete set of allocation vectors for a batch of N SFC requests is denoted by $\mathcal{A} = \{\alpha_1, \alpha_2, \dots, \alpha_N\} \in \Omega$, where $\Omega = \prod_{i=1}^N V^{|K_i|}$ is the generalized cartesian product of allocation vectors. Specifically, $\Omega = \{(\alpha_1, \dots, \alpha_N) \mid \alpha_i \in V^{|K_i|}, \forall i \in \{1, \dots, N\}\}$ defines the feasible solution space comprising tuples of allocation vectors over heterogeneous domains $V^{|K_1|}, V^{|K_2|}, \dots, V^{|K_N|}$, corresponding to the variable number of VNFs across the N SFC requests.

The communication latency for an SFC request f_i , given the allocation α_i , is modeled as $\Gamma(\alpha_i) = \sum_{(k_m, k_n) \in L_i} l(\alpha_{i,m}, \alpha_{i,n})$, where $l(\alpha_{i,m}, \alpha_{i,n})$ denotes the shortest-path transmission delay between VNF deployment locations $\alpha_{i,m}$ and $\alpha_{i,n}$. We define the processing latency for an SFC request f_i as $P(\alpha_i) = \sum_{j=1}^{|K_i|} (P^c(\alpha_{i,j}) + P^q(\alpha_{i,j}))$,

where $P^c(\alpha_{i,j})$ and $P^q(\alpha_{i,j})$ represent the computing delay and queuing delay waiting for processing of the j -th VNF at its assigned node, respectively. The total SFC delay for a request f_i is then $T(\alpha_i) = P(\alpha_i) + \Gamma(\alpha_i)$, comprising the communication, processing, and queuing delays. We therefore define the *total delay* for a batch of N requests as $T(A) = \sum_{\alpha_i \in A} T(\alpha_i)$, representing the aggregate delay across all SFCs in the batch.

The energy consumption at time t given an allocation α_i is modeled as

$$P_{\alpha_i}(t) = \sum_{v \in V} \omega_v(\alpha_i) \cdot P_v^{\text{Pr}}(t) + \sum_{v \in V} (1 - \omega_v(\alpha_i)) \cdot P_v^{\text{N}}(t) \quad (1)$$

where $\omega_v(\alpha_i) \in \{0, 1\}$ indicates whether node v is active under allocation α_i . The active processing energy consumption is defined as $P_v^{\text{Pr}}(t) = P_v^{\text{N}}(t) + \beta_v(t) \cdot (P_v^{\text{max}}(t) - P_v^{\text{N}}(t))$, with $P_v^{\text{N}}(t)$ representing baseline energy usage and $P_v^{\text{max}}(t)$ the energy drawn under full utilization [16]. The utilization factor $\beta_v(t)$ according to [13] is given by

$$\beta_v(t) = w_p \cdot \frac{x_{v_f}^k(t) \cdot \nu_f^k(t) \cdot \sigma_v^p}{C_v^p} + w_u \cdot \frac{x_{v_f}^k(t) \cdot \nu_f^k(t) \cdot \sigma_v^u}{C_v^u} \quad (2)$$

where $x_{v_f}^k(t)$ indicates whether flow f^k is processed on node v , $\nu_f^k(t)$ is the flow's processing rate, σ_v^p and σ_v^u are the resource demands per unit of computing and storage capacity, respectively, and C_v^p , C_v^u are the corresponding available capacities. The weights w_p and w_u reflect the relative importance of CPU and storage utilization. Let us define $P_v^{\tau}(t)$ as the transmission energy required to migrate VNF k from one node to another. We therefore define the *total energy consumption* for a batch of N requests as $P(A) = \sum_{\alpha_i \in A} P_{\alpha_i}(t)$, representing the cumulative energy usage induced by the current allocation across all requests in the batch.

The VNF migration cost comprises both time and energy components. The total migration time for a VNF $k \in K_i$ is defined as $t_k = \sum_{(i,j) \in l_k} \frac{M_k}{B_{ij}}$, which represents the sum of the transmission times required to transfer the VNFs' state size M_k over each physical link (i, j) with bandwidth B_{ij} along the shortest path l_k from the VNFs' current deployment location to its new candidate node. The associated energy consumption to migrate VNF k along the shortest path l_k is $e_k = \sum_{(i,j) \in l_k} P_v^{\tau}(t)$. We then define the SFC migration cost $M(\alpha_i) = \sum_{k \in K_i} \lambda \cdot t_k + (1 - \lambda) \cdot e_k$ as the sum of all VNF migration cost, where $\lambda \in [0, 1]$ is a tunable parameter used to balance the trade-off between time and energy costs, expressing different units as percentages. We therefore define the *total migration cost* for a batch of N requests as $M(A) = \sum_{\alpha_i \in A} M(\alpha_i)$, representing the aggregate migration costs across all SFCs in the batch.

Let us define n_v^k as the CPU cycles per second required by a VNF k when deployed on a physical node $v \in V$. Similarly, let b_{ij}^{kl} denote the bandwidth consumed by the logical link (k, l) of an SFC when mapped onto the physical link $(i, j) \in E$. We assume a total of M SFC requests arrive over time. To support scalable orchestration, we divide these into m batches, each

consisting of N concurrent requests, such that $m = M/N$. In each batch, VNFs are allocated jointly for the N SFCs.

Given that the optimization problem is multi-objective, we define $\beta = (\beta_1, \beta_2, \beta_3)$ as the weight vector balancing the trade-offs between aggregated delay, energy consumption, and migration cost. Recall that $T(A)$, $P(A)$, and $M(A)$ denote the total delay, energy consumption, and migration cost, respectively, aggregated over the batch of N SFC requests under allocation A . The objective is to minimize the scalarized cost function $\beta^\top C(A)$, where $C(A) = (T, P, M)(A)$ captures the three cost components. The goal of the optimization problem is to determine the optimal allocation $A^* \in \Omega$ for a batch of N SFC requests such that system-wide resource utilization is efficient and constraint-satisfying, as shown in Equation 3.

$$\underset{A \in \Omega}{\text{minimize}} \quad \overbrace{\beta_1 \cdot T(A)}^{\text{Total SFC delay}} + \overbrace{\beta_2 \cdot P(A)}^{\text{Energy consumption}} + \overbrace{\beta_3 \cdot M(A)}^{\text{Migration cost}} \quad (3)$$

subject to

$$\sum_{r=1}^m \sum_{i \in [N]} \sum_{k \in K_i} n_v^k \leq C_v, \quad \forall v \in V \quad (3a)$$

$$\sum_{r=1}^m \sum_{i \in [N]} \sum_{(k,l) \in L_i} b_{ij}^{kl} \leq B_{ij}, \quad \forall (i, j) \in E \quad (3b)$$

$$T(\alpha_i) \leq D_i^{\text{max}}, \quad \forall i \in \{1, \dots, N\} \quad (3c)$$

Constraint 3a ensures that for each node in the network, the total processing demand of all VNFs assigned to that node does not exceed its available computational capacity. Constraint 3b ensures that for each logical link, the combined bandwidth requirements of all SFC requests do not exceed the link's available bandwidth capacity. Constraint 3c ensures that for each SFC request, the aggregate processing and communication delay does not exceed its E2E delay tolerance required for successful service completion.

IV. MULTI-AGENT DEEP REINFORCEMENT LEARNING FOR DISTRIBUTED SFC DEPLOYMENT AND MIGRATION

We present *MARC-6G*, a distributed cooperative MARL framework that orchestrates VNF deployment and migration in dynamic networks. We redefine the optimization equation 1 in the context of the Multi Agent Reinforcement learning (MARL) problem, where each agent operates under a Partially Observable Markov Decision Process (POMDP), observing a local portion of the CCF, acting independently, and exchanging state, action, and reward to learn a joint policy that anticipates future conditions. The policy selects physical nodes for VNFs by accounting for inter-VNF delay, deployment cost, SLA constraints, congestion, energy, and migration cost. Agents refine strategies from experience and peer sharing, enabling scalable management of heterogeneous, time-varying workloads.

The broader system-wide objective is to jointly learn a set of optimal policies, denoted as $\pi^* = \{\pi_1^*, \dots, \pi_N^*\}$, which collectively maximize the performance of all agents in a cooperative manner.

A. Modeling VNF Deployment as Markov Decision Process

We model the distributed SFC deployment and migration task as a cooperative MARL problem, where the detailed formulation is expressed in terms of the joint state space S_t , joint action space A_t , and joint reward function R_t at a given time t , as described below.

- 1) **Joint State Space** S_t describes the current situation of each agent in the environment. The global state at time t is the collection of local states observed by each agent. The individual state of agent i is $S_t^i = (V_{i,t}^{K_i}, V_{i,t}^{K_2}, \dots, L_{V_{i,t},h}^{K_i}, G^{\text{topo}}, M, f_i)$ where $V_{i,t}^{K_i}$ indicates that VNF K_i is deployed on physical node V_i at time t , and $L_{V_{i,t},h}^{K_i}$ represents the link delay between node V_i (hosting K_i) and node V_h (hosting $|K_i|$). The state information also includes the observed network topology (G^{topo}), the number of SFC requests in the queue M , and the characteristics of the SFC request f_i .
- 2) **Joint Action Space** A_t explores optimal physical-node placements for hosting VNFs. Each action corresponds to selecting a sequence of physical nodes that meet the performance requirements of incoming SFC requests. At each time step t , each agent i selects a sequence of physical nodes to deploy the VNFs of its SFC. The action of agent i is defined as: $\alpha_t^i = (\alpha_{i,1}, \alpha_{i,2}, \dots, \alpha_{i,|K_i|}) \in V^{|K_i|}$ where each element $\alpha_{i,K_i} \in V$ represents the selected physical node on which the i -th VNF for the SFC request f_i is deployed.
- 3) **Joint Reward Function** R_t assigns a numerical score to each agent's decision on the network performance. The agent evaluates each physical node's placement by interpreting patterns it learns from the environment. At time t , the reward for agent i according to Equation 3 is given as follows. $R_t^i = -\sum_{t=0}^T \gamma^t \cdot (\beta_1 \cdot T(A) + \beta_2 \cdot P(A) + \beta_3 \cdot M(A))$ where T is the maximum time-step at which an agent learn optimal deployment policies. The collective reward is then defined as: $R = \sum_{i=1}^N R_t^i$, where α_t is the joint action across all agents.

B. Multi-Agent Proximal Policy Optimization for Autonomous VNF deployment

In Multi Agent Proximal Policy Optimization (MAPPO), distributed learning via experience sharing employs separate policy networks, each parameterized by π_{θ_i} , and separate value networks, each parameterized by V_{ϕ_i} . The RL agents exchange experiences through their value networks and policy networks to learn coordinated deployment policies. Each agent i observes a local state s_t^i (e.g., a subset of the network topology, current VNF placements, resource utilization, and KPI requirements for incoming traffic) and selects a deployment action $a_t^i \sim \pi_{\theta_i}(a_t^i | s_t^i)$. After agent i executes its action, it receives a reward r_{t+1}^i , which is weighted by β_i and exchanged with its neighboring agents. Based on this reward, each agent i updates its value function $V_{\phi_i}(s_t^i)$ and the policy network parameters θ_i .

In each training epoch, agents also share their latest policy $\pi_{\theta_i}(\cdot | s_t^i)$ with neighbors, enabling them to anticipate others' behaviors and coordinate VNF placements across the network. Thus, by exchanging weighted reward signals $\beta_i r_{t+1}^i$ (i.e., experience of an agent), MAPPO ensures that each agent's value network gains a more global perspective on performance.

Algorithm 1: MARC-6G Operation

Input: $F = (f_1, f_2, \dots, f_N)$, $T_{\max}$, G , $l(i, j)$, M , N
// Initialize policy, value, replay buffer
1 **Initialization:** ϕ_0^i, π_0^i, D_t
// Randomly place Initially VNF
2 $S_t^i = (V_{i,t}^{K_i}, V_{i,t}^{K_2}, \dots, L_{V_{i,t},h}^{K_i})$
// Initialize the reward to zero
3 $R_t^i \leftarrow 0$
4 **for** $t \in T_{\max}$ **do**
5 **for** $i \in N$ **do**
6 **// Each agent selects one SFC request**
7 $f_i \leftarrow \text{Sample}(M, i)$
8 $\bar{c} \leftarrow \text{MeasureCPUrequirement}(f_i)$
9 $\bar{b} \leftarrow \text{MeasureBandwidth}(f_i)$
10 $\bar{l} \leftarrow \text{MeasureLatency}(f_i)$
11 $\bar{q} \leftarrow \text{MonitorLinkquality}(G^{\text{topo}})$
12 **// Translate f_i & number of SFC requests in a queue as state**
13 $S_t^i \leftarrow (\bar{l}, \bar{b}, \bar{c}, \bar{q}, G^{\text{topo}}, M)$
14 **// Execute $A_t^i \sim \pi_{\theta_i}$ based on the current policy**
15 $S_t^i \xrightarrow{A_t^i, \pi_{\theta_i}(A_t^i | S_t^i, A_t^{i-1})} S_{t+1}^i, R_{t+1}^i$
16 **// Deploy K_i new location**
17 $S_{t+1}^i \leftarrow V_t^{K_i}$
18 **// Store each s_t^i, a_t^i, R_t^i in replay buffer**
19 $D_t \leftarrow \{s_0^i, a_0^i, r_0^i, \dots, s_t^i, a_t^i, r_t^i\}$
20 **if** $l(f_i) < l(i, j)$ **then**
21 $A_t^i \sim \pi_{\theta_i}(A_t^i | S_t^i, A_t^{i-1}) \xrightarrow{\text{Execute}} R_{t+1}^i, S_{t+1}^i$
22 **// Each agent calculates advantage estimate**
23 $A^{\pi_{\theta_i}}(s_t, a_t) \leftarrow Q^i(s, a) - V_{\phi_i}^i(s)$
24 **// Update θ_i and ϕ_i parameters**
25 $\theta_{t+1}^i = \arg \max_{\theta^i} \frac{1}{|\mathcal{D}_t|T} \sum_{\tau \in \mathcal{D}_t} \sum_{t=0}^T$
26 $\min \left(\frac{\pi_{\theta_t^i}(a_t | s_t)}{\pi_{\theta_t^i}(a_t | s_t)} A^{\pi_{\theta_t^i}}(s_t, a_t), g(\epsilon, A^{\pi_{\theta_t^i}}(s_t, a_t)) \right)$
27 $\phi_{t+1}^i =$
28 $\arg \min_{\phi} \frac{1}{|\mathcal{D}_t|} \sum_{\tau \in \mathcal{D}_t} \sum_{t=0}^T (V_{\phi}(s_t) - R_t)^2$
29 **else**
30 **// Relocate VNF**
31 $V_{t+1}^{K_i} \leftarrow V_t^{K_i}$
32 **// Assign VNF K^i optimally**
33 $K_i \xrightarrow{\text{Deploy}} V_t^{K_i}$
34 **return** $\phi_i(s_t), \pi_{\theta_i}(s_t)$

C. MARC-6G Algorithm

The step-by-step workflow of Algorithm 1 is described below in detail. The inputs to the MARC-6G algorithm are the physical network topology G^{topo} , the corresponding resource capacities (e.g., link bandwidth, link delay, node CPU), and a set of SFC requests f_i with their characteristics. Each agent selects a single SFC request from the pool M , based on its traffic

arrival time Λ_i and E2E delay requirement $D_i^{\max}$ (lines 5-7). Each agent constructs the state s_t by measuring the SFC request KPIs f_i (bandwidth, CPU, latency) and the current physical resource availability (lines 7-12). Next each agent execute the deployment action $a_t \sim \pi_\theta(s_t)$ by considering the deployment action of others, following current policy π_θ , update the system to state s_{t+1} , and set the new VNF placement $S_{t+1} = V_i^k$ store s_t^i, a_t^i, r_t^i trajectory in replay buffer D_t (lines 13-14). Then the requirements of $l(f_i)$ are compared with available infrastructure resources: $l(f_i) < l(i, j)$ for the given allocation vector. The reward R_t and the advantage $A^{\pi_{\theta_t}}(s_t, a_t)$ are then computed (lines 11-15). Then update θ_{t+1}^i (the policy) and ϕ_{t+1}^i (value parameters), and repeat this iteration until the optimal policy is developed (lines 14-21). The decision about the SFC request deployment or migration to another node (lines 21-22). Finally, return the value parameter $\phi_i(s_t)$ and the policy parameter $\pi_{\theta_i}(s_t)$ (line 23).

V. PERFORMANCE EVALUATION

A. Experimental Setup

We performed the simulation experiments using a custom simulator developed in Python with the *NetworkX* [17] to generate USA NET [18] network topologies that represent the underlying network infrastructure. For the MARC-6G implementation, we employ the open-source libraries *Gymnasium v1.0* and *RLlib v2.4.0* to train and evaluate RL agents within a custom environment. Given the variability of incoming traffic, each SFC request f_i is managed in a way that preserves better link quality. Multiple PPO agents operate in parallel, continuously monitoring network conditions and adjusting VNF placements in response to traffic patterns and system dynamics. We model both the state space S and the action space A as multi-discrete, making PPO a suitable choice due to its robustness, stability, and convergence properties, and update its policy online in discrete control tasks.

B. Baselines and Evaluation Metrics

The performance of MARC-6G, compared with centralized orchestration from the previous work [19] and baseline greedy-based VNF allocation. The Greedy allocator, deploying VNFs immediately without considering long-term impacts, often leads to suboptimal resource utilization. Centralized orchestrator Single Agent Proximal Policy Optimization (SAPPO), suffers from scalability limitations due to its reliance on a single global controller, preventing real-time adaptability in large-scale, dynamic network environments.

MARC-6G is evaluated using: (i) *Reward*, the weighted objective function that combines total E2E delay, energy consumption, and migration cost (Equation 3); (ii) *Number of Accepted Requests*, the fraction of VNF requests successfully deployed with the required performance relative to all incoming requests; (iii) *Energy Consumption*, highlighting energy-efficient VNF placement that minimizes usage by aggregating workloads from underutilized nodes onto a minimal set of active servers in real time; and (iv) *Migration Cost*, include the computational,

energy, and bandwidth overhead incurred when migrating VNF instances to optimize resource utilization and service quality.

C. Discussion and Simulation Results

Figure 3a compares learning curves for MARC-6G, SAPPO, and a greedy (non-learning) allocator: rewards fluctuate during exploration and SAPPO leads early (no coordination overhead), but once a shared state emerges, MARC-6G's multi-agent cooperation accelerates learning past SAPPO, ultimately outperforming both SAPPO and the greedy baseline with more stable, efficient VNF placements.

Figure 3b shows that across the SFC types in [3] (ID4.0, MIoT, CG, AR, VS), MARC-6G with five agents consistently outperforms SAPPO and the greedy allocator in terms of the number of accepted requests. For ID4.0, SAPPO and MARC-6G are comparable because ID4.0 has few VNFs, reducing contention. MARC-6G deploys SFCs concurrently with five agents, whereas SAPPO and the greedy baseline place one per step; the greedy is further hindered by random VNF placement.

Figure 3c shows energy consumption rising with the number of devices because VNFs are spread across many servers without accounting for over- or underprovisioning. MARC-6G lowers energy consumption by up to 12.5% and 39.2% compared to SAPPO and greedy across device scales by learning utilization-aware, energy-efficient VNF placements.

Figure 4a shows that as the number of physical devices grows, migration cost decreases: a larger pool of nodes enables optimal nearby node selection, and MARC-6G's multi-agent monitoring further minimizes cost by up to 34% and 41.25% compared to SAPPO and greedy approaches, respectively. Figure 4b reports E2E delay for 40/70/100-node networks under concurrent SFC loads of 3, 6, 9, and 12; delay drops across all loads as node count grows, indicating that MARC-6G learns scalable, resource-aware deployments that minimize latency even under heavier traffic.

Figure 4c illustrates the scalability of MARC-6G: as the number of SFC requests and hence agents increases proportionally, the end-to-end delay decreases, since each agent, managing only a segment of the network, and deploy multiple requests concurrently.

VI. CONCLUSION

This paper addresses distributed, context-aware VNF placement and migration under time-varying traffic for 6G networks, with the objective of minimizing end-to-end delay, energy consumption, and migration cost. We present *MARC-6G*, a MARL framework for concurrent SFC deployment that adapts online to network dynamics, placing and migrating VNFs to meet the strict delay requirements of fluctuating applications. Unlike centralized orchestration, characterized by slow, global state collection as request volume and topology size grow, MARC-6G monitors the local portion of the CCF, shares experience among agents, and updates policies in real time. Experimental results show that MARC-6G has higher request acceptance, better energy efficiency, lower migration cost, and improved scalability compared to baseline approaches.

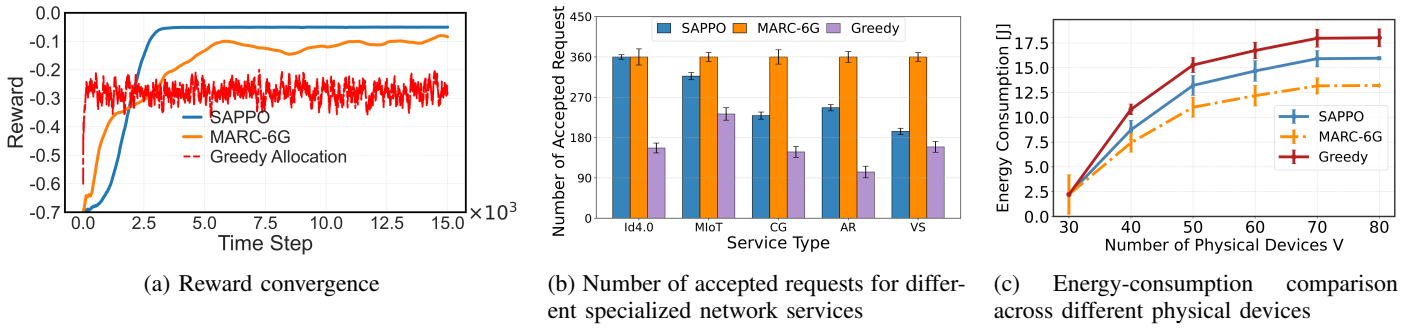


Fig. 3: Learning performance of MARC-6G: accepted SFC requests and energy consumption across the number of devices, compared with the baseline methods.

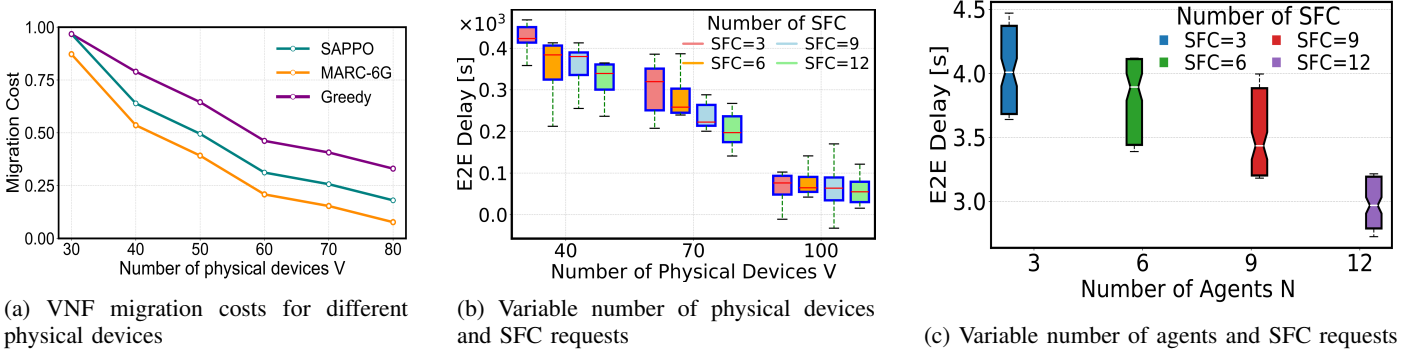


Fig. 4: Analysis of migration cost and scalability under varying SFC request loads and numbers of agents, and their impact on E2E delay.

ACKNOWLEDGMENT

This work is funded by the SNS-JU 6G Cloud project under the EU Horizon Europe programme (Grant Agreement No. 101139073).

REFERENCES

- [1] C. Campolo, A. Iera, and A. Molinaro, "Network for distributed intelligence: A survey and future perspectives," *IEEE Access*, vol. 11, pp. 52 840–52 861, 2023.
- [2] I. Avgouleas, D. Yuan, N. Pappas, and V. Angelakis, "Virtual network functions scheduling under delay-weighted pricing," *IEEE Networking Letters*, vol. 1, no. 4, pp. 160–163, 2019.
- [3] J. M. Ziazet, B. Jaumard, H. Duong, P. Khoshabi, and E. Janulewicz, "A dynamic traffic generator for elastic 5g network slicing," in *2022 IEEE international symposium on measurements & networking (M&N)*. IEEE, 2022, pp. 1–6.
- [4] S. Long, B. Liu, H. Gao, X. Su, and X. Xu, "Deep reinforcement learning-based sfc deployment scheme for 6g iot scenario," in *2023 IEEE Symposium on Computers and Communications (ISCC)*. IEEE, 2023, pp. 1189–1192.
- [5] N. Toumi, M. Bagaa, and A. Ksentini, "Machine learning for service migration: a survey," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 3, pp. 1991–2020, 2023.
- [6] T. Haga, "Orchestration of networking processes," 2007.
- [7] T. Mai, H. Yao, N. Zhang, W. He, D. Guo, and M. Guizani, "Transfer reinforcement learning aided distributed network slicing optimization in industrial iot," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 6, pp. 4308–4316, 2022.
- [8] M. A. Onsu, P. Lohan, B. Kantarci, E. Janulewicz, and S. Slobodrian, "Unlocking reconfigurability for deep reinforcement learning in sfc provisioning," *IEEE Networking Letters*, 2024.
- [9] Q. Liu, L. Tang, T. Wu, and Q. Chen, "Deep reinforcement learning for resource demand prediction and virtual function network migration in digital twin network," *IEEE Internet of Things Journal*, vol. 10, no. 21, pp. 19 102–19 116, 2023.
- [10] L. Tang, Z. Li, J. Li, D. Fang, L. Li, and Q. Chen, "Dt-assisted vnf migration in sdn/nfv-enabled iot networks via multiagent deep reinforcement learning," *IEEE Internet of Things Journal*, vol. 11, no. 14, pp. 25 294–25 315, 2024.
- [11] B. R. Tanuboddi, G. Gad, Z. M. Fadlullah, and M. M. Fouda, "Optimizing vnf migration in b5g core networks: A machine learning approach," in *2024 International Conference on Smart Applications, Communications and Networking (SmartNets)*. IEEE, 2024, pp. 1–5.
- [12] R. Chen, H. Lu, Y. Lu, and J. Liu, "Msdf: A deep reinforcement learning framework for service function chain migration," in *2020 IEEE Wireless communications and networking conference (WCNC)*. IEEE, 2020, pp. 1–6.
- [13] J. Chen, J. Chen, K. Guo, R. Hu, T. Zou, J. Zhu, H. Zhang, and J. Liu, "Fault tolerance oriented sfc optimization in sdn/nfv-enabled cloud environment based on deep reinforcement learning," *IEEE Transactions on Cloud Computing*, vol. 12, no. 1, pp. 200–218, 2024.
- [14] Y. Zhang, R. Wang, J. Hao, Q. Wu, Y. Teng, P. Wang, and D. Niyato, "Service function chain deployment with vnf-dependent software migration in multi-domain networks," *IEEE Transactions on Mobile Computing*, pp. 1–18, 2024.
- [15] 6G-Cloud, "D2.2 - Initial Results on Architecture, Service Interfaces and AI/ML," 6G-Cloud Consortium, Deliverable D2.2, 2025.
- [16] J. A. Aroca, A. Chatzipapas, A. F. Anta, and V. Mancuso, "A measurement-based characterization of the energy consumption in data center servers," *IEEE Journal on selected areas in communications*, vol. 33, no. 12, pp. 2863–2877, 2015.
- [17] NetworkX Developers, "Networkx," <https://networkx.org/>, 2025, accessed: 31 May 2025.
- [18] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson, "Measuring isp topologies with rocketfuel," *IEEE/ACM Transactions on networking*, vol. 12, no. 1, pp. 2–16, 2004.
- [19] S. F. Wassie, A. Di Maio, and T. Braun, "Deep reinforcement learning for context-aware online service function chain deployment and migration over 6g networks," in *Proceedings of the 40th ACM/SIGAPP Symposium on Applied Computing*, 2025, pp. 1361–1370.