

# ORIENT: A Priority-Aware Energy-Efficient Approach for Latency-Sensitive Applications in 6G

Masoud Shokrnezhad<sup>1</sup>, and Tarik Taleb<sup>1,2</sup>

<sup>1</sup> *Oulu University, Oulu, Finland; {masoud.shokrnezhad, tarik.taleb}@oulu.fi*

<sup>2</sup> *Ruhr University Bochum, Bochum, Germany; tarik.taleb@rub.de*

**Abstract**—Anticipation for 6G’s arrival comes with growing concerns about increased energy consumption in computing and networking. The expected surge in connected devices and resource-demanding applications presents unprecedented challenges for energy resources. While sustainable resource allocation strategies have been discussed in the past, these efforts have primarily focused on single-domain orchestration or ignored the unique requirements posed by 6G. To address this gap, we investigate the joint problem of service instance placement and assignment, path selection, and request prioritization, dubbed PIRA. The objective function is to maximize the system’s overall profit as a function of the number of concurrently supported requests while simultaneously minimizing energy consumption over an extended period of time. In addition, end-to-end latency requirements and resource capacity constraints are considered for computing and networking resources, where queuing theory is utilized to estimate the Age of Information (AoI) for requests. After formulating the problem in a non-linear fashion, we prove its NP-hardness and propose a method, denoted ORIENT. This method is based on the Double Dueling Deep Q-Learning (D3QL) mechanism and leverages Graph Neural Networks (GNNs) for state encoding. Extensive numerical simulations demonstrate that ORIENT yields near-optimal solutions for varying system sizes and request counts.

**Index Terms**—6G, Resource Allocation, Energy Consumption, Service Placement and Assignment, Path Selection, Prioritization, E2E Latency, Age of Information (AoI), Reinforcement Learning, Q-Learning, and Graph Neural Networks (GNNs).

## I. INTRODUCTION

The advent of the 6th generation of telecommunication systems (6G) signifies a pivotal era marked by unparalleled connectivity and technological advancements. With ultra-low End-to-End (E2E) latency (less than 1 millisecond), exceeding 1 terabit per second peak data rates, and ultra-high reliability surpassing 99.99999% [1], 6G promises to revolutionize industries such as holographic telepresence utilizing extended reality [2], dynamic metaverse empowered by semantic communications [3], and quantum networking [4]. However, achieving these capabilities raises substantial energy consumption concerns for both computing and networking resources. Presently, these resources consume around 200 terawatt-hours of electricity annually, approximately 1% of the global total [5]. Many quality-sensitive applications may require uploading up to 50% of data to computing facilities for processing [6], adding even more strain to computing and networking resources. Moreover, the projected surge in 6G-connected devices and global data exacerbates the energy consumption challenge, underscoring the need for sustainable solutions.

In order to realize a 6G-enabled future, it may be necessary to create novel resource orchestration mechanisms to address impending energy challenges. The subject has been extensively studied in the literature. Xuan *et al.* [7] addressed the Service Function Chaining (SFC) problem with the objective of minimizing energy consumption by proposing an algorithm based on multi-agent Reinforcement Learning (RL) and a self-adaptive division strategy. Solozabal *et al.* [8] investigated the same problem and proposed a single-agent solution. Other authors have also examined the SFC problem. By proposing a sampling-based Markov approximation method, Pham *et al.* [9] solved the problem in an effort to minimize operational and traffic energy consumption. Santos *et al.* [10] developed two policy-aware RL algorithms based on actor-critic and proximal policy optimization to maximize availability while minimizing energy consumption. Reducing energy consumption was considered in the Service Function (SF) placement problem as well. Sasan *et al.* [11] presented a heuristic algorithm to tackle the joint problem of network slicing, path selection, and SF placement, with the objective of maximizing user acceptance while minimizing energy consumption. Farhodi [12] and He *et al.* [13] investigated a comparable problem and proposed RL-based solutions, taking into account the dynamic nature of service requests and overall cost considerations (including operation, deployment, and transmission), respectively.

While effective in specific contexts, the mentioned methods may not be suitable for 6G systems. These approaches prioritize energy efficiency over maximizing device support, whereas achieving an E2E efficient solution requires holistic management of computing and networking resources, considering the stringent Quality of Service (QoS) demands of 6G. Furthermore, certain studies overlook or oversimplify critical network parameters like latency, which contradicts the intricate and dynamic requirements of 6G systems. This paper addresses this gap by investigating the joint problem of allocating computing and networking resources (service instance placement and assignment, path selection, and request prioritization), termed PIRA. The objective is to optimize the system’s overall profit (as a function of supported concurrent requests) while minimizing energy consumption over time, accounting for E2E latency and resource capacity constraints. The  $M/M/1$  queuing model is employed to accurately evaluate request latency on compute nodes and network devices. To solve this problem, we propose ORIENT, an approach leveraging Double

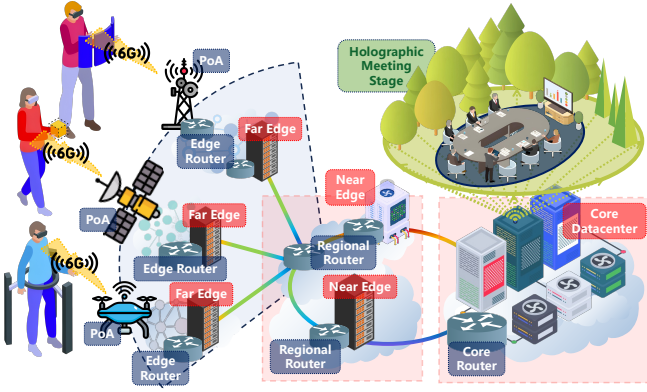


Figure 1. The system model, including network devices and distributed compute nodes facilitating holographic telepresence services for end users.

Dueling Deep Q-Learning (D3QL) reinforced by Graph Neural Networks (GNNs). This hybrid method effectively encodes the system state and facilitates the identification of near-optimal solutions.

The remainder of this paper is organized as follows. Section II introduces the system model. PIRA is defined and formulated in Section III, and ORIENT is presented in Section IV. Finally, numerical results are illustrated in Section V, followed by concluding remarks in Section VI.

## II. SYSTEM MODEL

As shown in Fig. 1, the following is an explanation of the two main components of the system: resources and requests.

### A. Resources

The 6G system examined in this paper is an integrated infrastructure of computing and networking resources comprised of  $\mathcal{N}$  network devices and  $\mathcal{V}$  compute nodes (radio resources are excluded) [14].  $\mathbb{N} = \{n | 0 \leq n \leq \mathcal{N}\}$  is the set of network devices, and  $\mathbb{V} = \{v | 0 \leq v \leq \mathcal{V}\}$  denotes the set of compute nodes. Compute nodes are connected through network devices via  $\mathcal{P}$  paths, the set of which is denoted by  $\mathbb{P} = \{p | 0 \leq p \leq \mathcal{P}\}$ , and the immediate network device of compute node  $v$  is indicated by  $n_v$ . Each path  $p$  contains a set of links  $\mathbb{L}_p \subset \mathbb{L}$ , where  $\mathbb{L} = \{l : (n, n') | n, n' \in \mathbb{N}\}$  is the set of all network links, and  $\mathcal{L}$  is its size. Network devices and compute nodes are priority-aware, i.e.,  $\mathbb{K} = \{k | 0 \leq k \leq \mathcal{K}\}$  is regarded as the set of permissible priority levels (where lower levels indicate higher priorities), and the resources in both domains are virtually partitioned, isolated, and guaranteed for each priority level  $k$ . Note that higher priorities receive a larger share of available resources than lower priorities.

To evaluate the performance of allocated resources, we will employ the  $M/M/1$  queuing model for each priority level on network devices and compute nodes assuming that this theory's stability requirements are met and that all queues are independent. The service rate allocated to priority level  $k$  on network device  $n$  is  $\widehat{B}_{n,k}$ , and those packets leaving this queue will be forwarded through their corresponding link, let's call it  $l$ , allocating  $\widehat{B}_{l,k}$  bandwidth. Note that the overall capacity of this network device and link is constrained by

$\widehat{B}_n$  and  $\widehat{B}_l$ , respectively. Similarly, the requests of priority level  $k$  will be served on compute node  $v$  leveraging a queue with dedicated service rate  $\widehat{C}_{v,k}$ , and this node is equipped with computing resources limited to a predefined capacity threshold, dubbed  $\widehat{C}_v$ . In addition, the energy consumption for transmitting bandwidth units over network device  $n$  is  $\widehat{\mathcal{E}}_n$ , and compute node  $v$  consumes  $\widehat{\mathcal{E}}_v$  energy per capacity unit and  $\overline{\mathcal{E}}_v$  energy when its state changes (from the idle mode to the operation mode or vice versa).

### B. Requests

This paper investigates the system for  $\mathcal{T}$  time slots while a set of  $\mathcal{R}_t$  requests, denoted by  $\mathbb{R}_t = \{r | 0 \leq r \leq \mathcal{R}_t\}$ , arrives at time slot  $t \in \mathbb{T} = \{t | 1 \leq t \leq \mathcal{T}\}$ . The set of all requests is  $\mathbb{R} = \{\mathbb{R}_t | 1 \leq t \leq \mathcal{T}\}$ , and  $\mathcal{R}$  represents the number of all requests. Each request  $r$  enters the system through an edge network device, denoted by  $n_r$  and referred to as its Point of Arrival (PoA), and orders service  $s_r$  from the set of obtainable service instances, that is  $\mathbb{I} = \{i | 0 \leq i \leq \mathcal{I}\}$ , where instance  $i$  provides service  $s_i$ . In order to successfully fulfill each request, one instance of its target service must be replicated on one of the compute nodes in order to receive the request, process it, and return it to its entry point so that it can be delivered to the end user.  $\widehat{C}_i$  represents the maximum capacity of instance  $i$ . To fulfill each user's request, its QoS requirements must be met, including the minimum service capacity and network bandwidth, as well as the maximum tolerable E2E latency, denoted by  $\widehat{C}_r$ ,  $\widehat{B}_r$ , and  $\widehat{D}_r$ , respectively. Besides, the maximum permissible packet size for request  $r$  is  $\widehat{H}_r$ . If request  $r$  is successfully completed, the system will achieve a profit, that is  $\gamma_r$ . Note that the arrival rate for each queue is determined by a Poisson process and is assumed to be the sum of  $\widehat{C}_r$  (for compute queues) and  $\widehat{B}_r$  (for network queues) for all requests assigned to that queue, respectively.

## III. PROBLEM DEFINITION

This section discusses the joint problem of instance placement and assignment, request prioritization, and path selection for integrated compute-network infrastructures to maximize the overall profit of the system while minimizing its energy consumption. In this section, the constraints and objective function are formulated, followed by the problem statement as a Mixed-Integer Non-Linear Programming (MINLP) formulation and its complexity analysis.

### A. Instance Orchestration Constraints

Constraints C1-C6 assign requests to instances and place them on compute nodes while maintaining the capacity constraints of instances and compute nodes. Considering that  $\dot{I}_{r,i}^t$  is a binary variable whose value is 1 if request  $r$  is assigned to instance  $i$  at time slot  $t$ , C1 ensures that each request  $r$  is assigned to no more than one instance of its service for each time slot  $t$ . C2 defines a new binary variable,  $\dot{I}_i^t$ , which indicates that whether instance  $i$  is activated at time slot  $t$ . If  $\sum_{\mathbb{R}_t} \dot{I}_{r,i}^t$  is equal to or greater than 1 (i.e., at least one request is assigned to instance  $i$ ),  $(\sum_{\mathbb{R}_t} \dot{I}_{r,i}^t) / \mathcal{R}_t$  will be a

small number (between 0 and 1) and  $\sum_{\mathbb{R}_t} \tilde{I}_{r,i}^t$  will be a large number, so  $\hat{I}_i^t$  will be set to 1. Otherwise, both sides of the equation will equal 0, causing  $\hat{I}_i^t$  to also equal 0. C3 ensures that each activated instance is assigned to exactly one compute node, where  $\tilde{G}_{i,v}^t$  is a binary variable indicating the compute node of instance  $i$  at time slot  $t$ . Similar to C2, C4 defines  $\hat{G}_v^t$  as a binary variable to determine whether compute node  $v$  should be activated at time slot  $t$ . Then, it must be assured that assigned requests do not exceed the capacity limitations of instances and compute nodes (C5 and C6).

$$\sum_{\mathbb{I}|s_i=s_r} \tilde{I}_{r,i}^t \leq 1 \quad \forall t, r \in \mathbb{T}, \mathbb{R}_t \quad (\text{C1})$$

$$\frac{1}{\mathcal{R}_t} \cdot \sum_{\mathbb{R}_t} \tilde{I}_{r,i}^t \leq \hat{I}_i^t \leq \sum_{\mathbb{R}_t} \tilde{I}_{r,i}^t \quad \forall t, i \in \mathbb{T}, \mathbb{I} \quad (\text{C2})$$

$$\sum_{\mathbb{V}} \tilde{G}_{i,v}^t = \hat{I}_i^t \quad \forall t, i \in \mathbb{T}, \mathbb{I} \quad (\text{C3})$$

$$\frac{1}{\mathbb{I}} \cdot \sum_{\mathbb{I}} \tilde{G}_{i,v}^t \leq \hat{G}_v^t \leq \sum_{\mathbb{I}} \tilde{G}_{i,v}^t \quad \forall t, v \in \mathbb{T}, \mathbb{V} \quad (\text{C4})$$

$$\sum_{\mathbb{R}_t} \tilde{C}_r \cdot \tilde{I}_{r,i}^t \leq \hat{C}_i \quad \forall t, i \in \mathbb{T}, \mathbb{I} \quad (\text{C5})$$

$$\sum_{\mathbb{I}} \hat{C}_i \cdot \tilde{G}_{i,v}^t \leq \hat{C}_v \quad \forall v, t \in \mathbb{V}, \mathbb{T} \quad (\text{C6})$$

### B. Path Selection Constraints

Constraints C7-C9 ensure that an E2E path is selected for each request considering the capacity constraints of network links and the traffic pattern, where packets of each request enter the network through its PoA and, after visiting its assigned instance, are returned to the same PoA to be handed off to the corresponding end user. C7 determines the allocated path for each request  $r$ , ensuring that it originates and terminates at  $n_r$  and traverses the network device directly connected to the compute node hosting the instance assigned to the request. In this constraint,  $\tilde{f}_{r,p}^t$  is a binary variable that represents the assigned path of request  $r$  at time slot  $t$ . Finally, C8 and C9 maintain the maximum capacity of network links and devices.

$$\sum_{\mathbb{P}|n_r \& n_v \in \mathbb{P}} \tilde{f}_{r,p}^t = \tilde{I}_{r,i}^t \cdot \tilde{G}_{i,v}^t \quad \forall t, r, i, v \in \mathbb{T}, \mathbb{R}, \mathbb{I}, \mathbb{V} \quad (\text{C7})$$

$$\sum_{\mathbb{R}_t, \mathbb{P}|l \in \mathbb{L}_p} \tilde{B}_r \cdot \tilde{f}_{r,p}^t \leq \hat{B}_l \quad \forall t, l \in \mathbb{T}, \mathbb{L} \quad (\text{C8})$$

$$\sum_{\mathbb{R}_t, \mathbb{P}|n \in \mathbb{L}_p} \tilde{B}_r \cdot \tilde{f}_{r,p}^t \leq \hat{B}_n \quad \forall t, n \in \mathbb{T}, \mathbb{N} \quad (\text{C9})$$

### C. Request Prioritization Constraints

$$\sum_{\mathbb{K}} \tilde{Q}_{r,k}^t = \sum_{\mathbb{I}} \tilde{I}_{r,i}^t \quad \forall t, r \in \mathbb{T}, \mathbb{R}_t \quad (\text{C10})$$

$$\sum_{\mathbb{R}_t, \mathbb{I}} \tilde{C}_r \cdot \tilde{Q}_{r,k}^t \cdot \tilde{I}_{r,i}^t \cdot \tilde{G}_{i,v}^t < \hat{C}_{v,k} \quad \forall t, k, v \in \mathbb{T}, \mathbb{K}, \mathbb{V} \quad (\text{C11})$$

$$\sum_{\mathbb{R}_t, \mathbb{P}|l \in \mathbb{L}_p} \tilde{B}_r \cdot \tilde{Q}_{r,k}^t \cdot \tilde{f}_{r,p}^t < \hat{B}_{l,k} \quad \forall t, k, l \in \mathbb{T}, \mathbb{K}, \mathbb{L} \quad (\text{C12})$$

$$\sum_{\mathbb{R}_t, \mathbb{P}|n \in \mathbb{L}_p} \tilde{B}_r \cdot \tilde{Q}_{r,k}^t \cdot \tilde{f}_{r,p}^t < \hat{B}_{n,k} \quad \forall t, k, n \in \mathbb{T}, \mathbb{K}, \mathbb{N} \quad (\text{C13})$$

To maintain integrity, it's crucial to prevent any overuse of resources allocated to each priority level. Given that  $\tilde{Q}_{r,k}^t$  is the priority of request  $r$  at time slot  $t$ , C10 promises that the request's priority is determined if an instance is assigned to serve it. Then, C11 to C13 satisfy the capacity constraints of priority queues on compute nodes and network resources.

### D. Latency Constraints

Each packet has to wait for three sources of latency through its request's assigned E2E path in the system: 1) the service latency experienced at the network devices included in the

path, 2) the transmission latency over the network links of the path, and 3) the service latency at the assigned compute node. Since the average latency of a packet in a  $M/M/1$  queue is equal to  $1/(\mu - \lambda)$  when the arrival rate is  $\lambda$  and the service rate is  $\mu$ , the average latency experienced by the packets of request  $r$  at network device  $n$  allocated to priority level  $k$  during time slot  $t$  can be calculated as C14. In this constraint, the numerator will be 0 for network devices and priority levels that have not been allocated to request  $r$ , causing  $\tilde{D}_{r,n,k}^t$  to equal 0. Otherwise, the numerator will be 1, and the latency will be determined following the adopted queuing theorem with the arrival rate of the queue set to the overall bandwidth of requests assigned to priority level  $k$  and traversing network device  $n$ . C15 follows the same logic to calculate the average latency of request  $r$  allocated to priority level  $k$  at time slot  $t$  on compute node  $v$ . C16 calculates the transmission latency of request  $r$  over link  $l$  at time slot  $t$ , considering its priority level and maximum packet size, if the link is part of the path assigned to the request. Otherwise, the latency will be 0. Finally, C17 determines the Age of Information (AoI), followed by C18, which ensures the maximum acceptable latency requirement of requests.

$$\tilde{D}_{r,n,k}^t = \frac{\sum_{\mathbb{P}|n \in \mathbb{L}_p} \tilde{Q}_{r,k}^t \cdot \tilde{f}_{r,p}^t}{\hat{B}_{n,k} - \sum_{\mathbb{R}_t, \mathbb{P}|n \in \mathbb{L}_p} \tilde{B}_{r'} \cdot \tilde{Q}_{r',k}^t \cdot \tilde{f}_{r',p}^t} \quad \forall t, r, k, n \in \mathbb{T}, \mathbb{R}_t, \mathbb{K}, \mathbb{N} \quad (\text{C14})$$

$$\tilde{D}_{r,v,k}^t = \frac{\sum_{\mathbb{I}} \tilde{Q}_{r,k}^t \cdot \tilde{I}_{r,i}^t \cdot \tilde{G}_{i,v}^t}{\hat{C}_{v,k} - \sum_{\mathbb{R}_t, \mathbb{I}} \tilde{C}_{r'} \cdot \tilde{I}_{r',i}^t \cdot \tilde{G}_{i,v}^t} \quad \forall t, r, k, v \in \mathbb{T}, \mathbb{R}_t, \mathbb{K}, \mathbb{V} \quad (\text{C15})$$

$$\tilde{D}_{r,l,k}^t = \frac{\hat{H}_r}{\hat{B}_{l,k}} \cdot \sum_{\mathbb{P}|l \in \mathbb{L}_p} \tilde{Q}_{r,k}^t \cdot \tilde{f}_{r,p}^t \quad \forall t, r, l, k \in \mathbb{T}, \mathbb{R}_t, \mathbb{L}, \mathbb{K} \quad (\text{C16})$$

$$\tilde{D}_r^t = \sum_{\mathbb{N}, \mathbb{K}, \mathbb{V}, \mathbb{L}} (\tilde{D}_{r,n,k}^t + \tilde{D}_{r,v,k}^t + \tilde{D}_{r,l,k}^t) \quad \forall t, r \in \mathbb{T}, \mathbb{R}_t \quad (\text{C17})$$

$$\tilde{D}_r^t \leq \tilde{D}_r \quad \forall t, r \in \mathbb{T}, \mathbb{R}_t \quad (\text{C18})$$

### E. Objective Function

The objective is to maximize the overall profit while minimizing the energy consumption of resources, that is:

$$\sum_{\mathbb{T}, \mathbb{R}_t} (\gamma_r \cdot \sum_{\mathbb{I}} \tilde{I}_{r,i}^t) - \alpha \cdot (\sum_{\mathbb{N}} \tilde{E}_n + \sum_{\mathbb{V}} \tilde{E}_v), \quad (\text{OF})$$

where  $\sum_{\mathbb{I}} \tilde{I}_{r,i}^t$  is 1 if request  $r$  is supported at time slot  $t$ ,  $\alpha$  is a small positive number, and  $\tilde{E}_v$  and  $\tilde{E}_n$  represent, respectively, the total energy consumption of compute node  $v$  and network device  $n$ . Note that  $\alpha$  must be set such that the total profit exceeds the total amount of energy consumed. Otherwise, supporting requests would result in a negative objective function value, and the only optimal solution would be to support no requests, making OF equal to 0. To determine  $\tilde{E}_n$ , where the only source of energy consumption is transmitting requests' data, the following equations are employed:

$$\tilde{E}_n = \tilde{E}_n \cdot \sum_{\mathbb{T}, \mathbb{R}_t, \mathbb{P}|n \in \mathbb{L}_p} \tilde{B}_r \cdot \tilde{f}_{r,p}^t \quad (1)$$

To calculate  $\tilde{E}_v$ , it should be noted that the energy consumed on each compute node has two primary sources: 1) the energy consumed to service each unit of requests' data, and 2) the energy consumed during booting up or shutting down the compute node. Consequently,  $\tilde{E}_v$  for each  $v \in \mathbb{V}$  is:

$$\tilde{\mathcal{E}}_v \cdot \sum_{\mathbb{T}, \mathbb{R}_t, \mathbb{I}} \tilde{\mathcal{C}}_r \cdot \tilde{\mathcal{I}}_{r,i}^t \cdot \tilde{\mathcal{G}}_{i,v}^t + \bar{\mathcal{E}}_v \cdot \sum_{\mathbb{T}|0 \leq t < \mathcal{T}} \hat{\mathcal{G}}_v^t \oplus \hat{\mathcal{G}}_v^{t+1} \quad (2)$$

In this equation,  $\tilde{\mathcal{I}}_{r,i}^t \cdot \tilde{\mathcal{G}}_{i,v}^t$  equals 1 if compute node  $v$  is selected as the host for request  $r$ ; therefore, the first part of the equation calculates the total energy consumption of compute node  $v$  to service its assigned requests. Assuming  $\hat{\mathcal{G}}_v^0$  equals 0,  $\hat{\mathcal{G}}_v^t \oplus \hat{\mathcal{G}}_v^{t+1}$  is equal to 1 in the second part if and only if  $\hat{\mathcal{G}}_v^t$  and  $\hat{\mathcal{G}}_v^{t+1}$  differ, representing a boot-up or shutdown for compute node  $v$ . Then, the second part demonstrates the total energy consumption of state transitions in compute node  $v$  within  $\mathbb{T}$ .

#### F. Problem

Considering the constraints and the objective function, the Problem of Integrated Resource Allocation (PIRA) is:

$$\text{PIRA: max OF s.t. C1 - C18.} \quad (3)$$

The optimal solution involves assigning requests with stringent latency requirements to high-priority queues and keeping those queues as empty as possible to minimize latency. Resource selection should aim to minimize activated resources and consider future requests to reduce energy consumption through fewer start-ups and shutdowns, improving energy efficiency.

#### G. Complexity Analysis

The problem defined in (3) is an extended version of the Multi-Dimensional Knapsack (MDK) problem. Assume the problem is relaxed and reformulated specifically for time slot  $t$  as the problem of maximizing profit and minimizing energy consumption while the only decision is to assign requests to instances concerning only their capacity constraints. Since the MDK problem is NP-hard and this relaxed version is an MDK problem with at least  $\mathcal{R}_t$  items and  $\mathcal{I}$  knapsacks, PIRA is at least as difficult as the MDK problem and is also NP-hard.

### IV. ORIENT

This section proposes an RL-based priORity-aware Energy-efficient laTency-sensitive resource allocation approach (ORIENT) to find near-optimal solutions for PIRA. Subsequently, the learning mechanism is elaborated upon, followed by an explanation of the agent's design, and concluding with a description of the algorithm.

#### A. Learning Mechanism

Given the continuous operation of the system defined in this paper and the recurring necessity for consistent resource allocation decisions at each time slot of PIRA, the adoption of RL presents itself as a viable means to enhance decision-making proficiency to solve it. Within the framework of RL, an agent undergoes a process of learning by means of trial and error at each step (here, time slot), with the primary aim of optimizing a specific decision-making problem. The system's designer defines a reward function in alignment with the objectives of the problem. By learning and following the optimal strategy derived from this reward function, the agent aims to maximize cumulative discounted rewards, regardless

of the initial state. Among various RL-based algorithms, Q-Learning stands out as widely acknowledged.

In Q-Learning, every state-action pair is associated with a numeric value referred to as the Q-value, where the agent selects the action with the maximum Q-value at each step. In Deep Q-Learning (DQL), a Deep Neural Network (DNN) serves as the approximator for these Q-values. In this arrangement, the state and action are presented as inputs, and the DNN-based Q-function encompassing all feasible actions, denoted by  $Q(s, \cdot; \mathcal{W})$ , is generated as the output and systematically updated over time according to the following equation:

$$\mathcal{W}^{t+1} = \mathcal{W}^t + \sigma [Y^t - Q(\mathcal{S}^t, a^t; \mathcal{W}^t)] \nabla_{\mathcal{W}^t} \cdot Q(\mathcal{S}^t, a^t; \mathcal{W}^t) \quad (4)$$

In this equation,  $\mathcal{W}$  is the set of DNN weights,  $\sigma$  is a scalar step size,  $\mathcal{S}^t$  and  $a^t$  are the agent's state and action at time slot  $t$ , and  $Y^t$  (also known as the target) shows the maximum value expected to be achieved by following  $a^t$  at  $\mathcal{S}^t$ .  $Y^t$  is the only variable that must be estimated in this equation, and in Double DQL (DDQL), where the selection and evaluation processes are decoupled, it can be expressed as follows:

$$Y^t = r^{t+1} + \gamma \hat{Q}(\mathcal{S}^{t+1}, a', \mathcal{W}^{t-}), \quad (5)$$

where  $r^{t+1}$  is the earned reward at time slot  $t+1$ ,  $\gamma \in [0, 1]$  is a discount factor that balances the importance of immediate and future rewards,  $a' = \operatorname{argmax}_{a \in \mathcal{A}} Q(\mathcal{S}^{t+1}, a, \mathcal{W}^t)$ , and  $\mathcal{A}$  is the set of actions. In this equation,  $\mathcal{W}$  represents the set of weights for the main  $Q$  and is updated in each step, whereas  $\mathcal{W}^-$  is for the target  $\hat{Q}$  and is replaced with the weights of the main network every  $t$  steps. In other words,  $\hat{Q}$  remains a periodic copy of  $Q$ .

Furthermore, we augment DDQL by incorporating the dueling concept introduced by Wang *et al.* [15]. Unlike conventional DDQL, which directly approximates Q-values using DNNs, this method initially computes separate estimators for state values ( $\psi$ ) and action advantages ( $\varphi$ ). Q-values are then derived from these estimators, as illustrated below:

$$Q(\mathcal{S}^t, a^t, \mathcal{W}^t) = \psi(\mathcal{S}^t, \mathcal{W}^t) \left( \varphi(\mathcal{S}^t, a^t, \mathcal{W}^t) - \frac{\Phi}{|\mathcal{A}|} \right) \quad (6)$$

where  $\Phi = \sum_{\mathcal{A}} \varphi(\mathcal{S}^t, a', \mathcal{W}^t)$ . The primary benefit is the ability to generalize learning across actions without modifying the learning algorithm, which improves policy evaluation in the presence of numerous actions with similar state values. As a result of combining the Dueling technique and DDQL, we can expect that the resultant D3QL agent will outperform its predecessors. To bolster the effectiveness and resilience of D3QL, observed transitions are archived in a memory bank known as the experience memory. The learning process entails randomly selecting transitions from this repository [16].

#### B. Agent Customization

The first step toward exploiting D3QL to solve PIRA is to define the agent's action space, state space, and reward.

*Action Space:* We define the action space as set  $\mathcal{A} = \{a : (i, v, p, k) | i, v, p, k \in \mathbb{I}, \mathbb{V}, \mathbb{P}, \mathbb{K}\}$ . During each time slot and for every request, a specific action must be executed to finalize the resource allocation pertaining to that request.

Tabelle I  
SIMULATION PARAMETERS.

Parameter	Value
number of priority levels	4
resource capacity bounds	$\sim \mathcal{U}\{250, 300\}$ mbps
Instance capacity bound	20 mbps
energy consumptions per capacity unit	$\sim \mathcal{U}\{10, 20\}$
energy consumptions per state transition	$\sim \mathcal{U}\{100, 200\}$
capacity requirement per request	$\sim \mathcal{U}\{4, 8\}$ mbps
bandwidth requirement per request	$\sim \mathcal{U}\{2, 10\}$ mbps
latency requirement per request	$\sim \mathcal{U}\{1, 3\}$ ms
packet size per request	1
profit per request ( $\gamma_r$ )	$\mathcal{U}\{5, 15\}$

*State Space:* For encoding the system's state, an architecture involving aggregation GNN layers is employed, constructing an aggregation sequence across all compute nodes, iteratively facilitating information exchange with neighboring nodes. Therefore, at time slot  $t$  when request  $r$  is on the verge of receiving service, the system state is denoted as  $\mathbf{S}^t(r) = \{\mathbf{S}_{\mathbb{V}}^t(r), \mathbf{S}_{\mathbb{P}}^t(r)\}$  and can be formally defined as:

$$\mathbf{S}_{\mathbb{V}}^t(r) = \left\{ \left[ \widehat{C}_{v,k}^t - \widetilde{C}_r \right]_{\mathbb{K}}, \left[ \widehat{D}_{r,v,k}^t - \widetilde{D}_r \right]_{\mathbb{K}}, \widetilde{\mathcal{E}}_v, \widetilde{\mathcal{E}}_v \cdot (1 - \dot{g}_v^t) \right\}_{\mathbb{V}}, \quad (7)$$

$$\mathbf{S}_{\mathbb{P}}^t(r) = \left\{ \left[ \wedge_{\mathbb{L}_p} - \widetilde{B}_r \right]_{\mathbb{K}}, \left[ \widehat{D}_r^t - \widehat{D}_{r,v,k}^t - \widetilde{D}_r \right]_{\mathbb{K}}, \widetilde{\mathcal{E}}_n \right\}_{\mathbb{P}}, \quad (8)$$

where  $\wedge_{\mathbb{L}_p} = \min_{n,l \in \mathbb{L}_p} \{\widehat{B}_{n,k}^t, \widehat{B}_{l,k}^t\}$ .  $\mathbf{S}_{\mathbb{V}}^t(r)$  and  $\mathbf{S}_{\mathbb{P}}^t(r)$  represent the embeddings of compute nodes and network paths, respectively, which function as inputs for the GNN layers. These embeddings encompass the remaining resource capacity and the anticipated latency when request  $r$  is allocated to them, as well as their associated energy consumption.

*Reward:* Since the agent is designated to maximize OF, the reward should be engineered to reinforce the support of high-profit requests while selecting resources with low energy consumption. This goal is satisfied in (9), that is:

$$r^{t+1} = \begin{cases} 0 & \text{otherwise} \\ \frac{\mathcal{M}}{\text{OF}(\mathbf{S}^t(r), a^t) - \min_{\mathcal{A}} \text{OF}(\mathbf{S}^t(r), a')} & r \text{ is met} \end{cases} \quad (9)$$

where  $\mathcal{M} = \max_{\mathcal{A}} \text{OF}(\mathbf{S}^t(r), a') - \min_{\mathcal{A}} \text{OF}(\mathbf{S}^t(r), a')$ ,  $\max/\min_{\mathcal{A}} \text{OF}(\mathbf{S}^t(r), a')$  is the maximum/minimum profit that can be achieved by allocating the available resources at time slot  $t$  to request  $r$  without considering any constraints or requirements, and  $\text{OF}(\mathbf{S}^t(r), a^t)$  is the profit of the allocation provided by the agent. If the action fails to meet the requirements of the request, it results in a reward of 0. Conversely, actions that yield greater profits correspond to higher rewards.

### C. ORIENT's Algorithm

ORIENT is detailed in Algorithm 1. In this algorithm,  $\epsilon'$  and  $\tilde{\epsilon}$  are small positive integers that control the  $\epsilon$ -greedy mechanism. During each time slot  $t$ , the agent receives notifications of new request arrivals ( $r$ ), and it computes the state based on request  $r$  requirements and the current system state. The action is then chosen using an  $\epsilon$ -greedy policy, which follows the evaluation function of the corresponding agent with probability  $(1 - \epsilon)$  and selects a random action with probability  $\epsilon$ .

### Algorithm 1: ORIENT

---

**Input:**  $\mathcal{T}$ ,  $\epsilon'$ , and  $\tilde{\epsilon}$

- 1  $\Omega \leftarrow \emptyset$ ,  $\mathcal{W} \leftarrow \mathbf{0}$ ,  $\mathcal{W}^- \leftarrow \mathbf{0}$ ,  $\epsilon \leftarrow 1$ ,  $memory \leftarrow \{\}$
- 2 **for each**  $t$  in  $[0 : \mathcal{T}]$  **do**
- 3     **if** new request  $r$  is arrived **then**
- 4         calculate  $\mathbf{S}^t(r) = \{\mathbf{S}_{\mathbb{V}}^t(r), \mathbf{S}_{\mathbb{P}}^t(r)\}$
- 5          $\zeta \leftarrow$  generate a random number from  $[0 : 1]$
- 6         **if**  $\zeta > \epsilon$  **then**
- 7              $a^t = (i, v, p, k) \leftarrow \text{argmax}_{\mathcal{A}} Q(\mathbf{S}^t, a', \mathcal{W}^t)$
- 8         **else**
- 9             select a random  $a^t = (i, v, p, k)$  from  $\mathcal{A}$
- 10         calculate  $r^{t+1}$
- 11         **if**  $r^{t+1} > 0$  **then**
- 12             Establish request  $r$  connection based on  $a^t$
- 13              $memory \leftarrow memory \cup \{(\mathbf{S}^t, a^t, r^{t+1})\}$
- 14             choose a batch of samples from  $memory$
- 15             train the agent
- 16             **if**  $\epsilon > \tilde{\epsilon}$  **then**
- 17                  $\epsilon \leftarrow \epsilon - \epsilon'$
- 18              $\Omega \leftarrow \Omega \cup \{(t, r, a^t)\}$
- 19 **return**  $\Omega$

---

Subsequently, the reward is calculated, and if it exceeds 0, it indicates that  $a^t$  is feasible and meets request  $r$ 's QoS requirements, enabling its connection based on  $a^t$  allocations. Finally, the experience memory is updated, samples are drawn from the memory bank, and the agent undergoes training. during the training process,  $\epsilon$  decreases from 1 to  $\tilde{\epsilon}$ . The algorithm yields  $\Omega$  as the history of allocations.

## V. PERFORMANCE EVALUATION

In this section, we present numerical results based on the system model parameters listed in Table I. Other parameters can be chosen arbitrarily so long as the logic outlined throughout the paper remains valid. To evaluate the efficiency of ORIENT, we conduct a comparative analysis with OPT, D3QL, and RND. OPT represents the optimal solution for PIRA, obtained through the use of CPLEX 12.10. D3QL, on the other hand, bears similarities to the method outlined in Algorithm 1, but employs exclusively simple linear layers in its DNN, without utilizing any GNNs. This approach forms the foundation for several related studies, including A-DDPG [13] and MDRL-SaDS [7], both of which are RL methods designed to enhance the utility of individual requests by considering factors such as resource cost, required bandwidth, and E2E path latency. Lastly, RND represents a random allocation strategy, where resources are allocated to active requests without considering any constraints.

The results are depicted in Fig. 2, where subfigures A and B represent the average energy consumption per request and total profit across various system sizes, with a constant of 300 active requests. Here, incrementing the system size entails the creation of a new system graph, incorporating  $\mathcal{N} + 1$  network devices and  $\mathcal{V} + 1$  compute nodes. Particularly, from

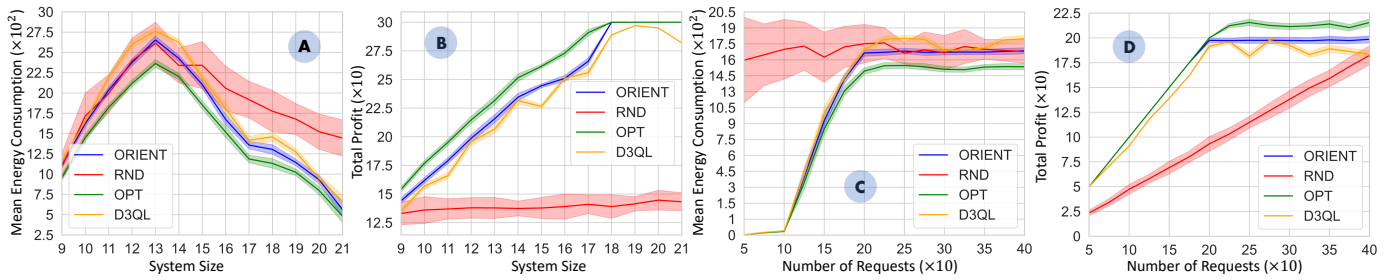


Figure 2. The mean energy consumption of supported requests and the total profit vs. the system size (A & B) and the number of all requests (C & D).

10 to 13, resources with a significant energy consumption are introduced into the graph. Between 14 and 17, resources with a moderate energy consumption are added to the graph, and the remaining resources included from 18 to 21 are characterized by low energy consumption. Subfigures C and D display similar quality metrics, but for different numbers of requests, while keeping the system size fixed at 12 with equal numbers of resources (4) from each level of energy consumption.

Within the subfigures, it is evident that OPT serves as an upper performance bound, while RND serves as the lower bound. Furthermore, when all resources exhibit high energy consumption rates or are fully occupied (with  $\mathcal{N} \leq 13$  in A and  $\mathcal{R} \geq 200$  in C), RND shows a similar energy consumption pattern to D3QL-based techniques, but its support rate is limited due to the absence of intelligence and feasibility checks. In contrast, ORIENT excels in both scenarios. As demonstrated in A and B, it achieves near-optimal results by prioritizing high-capacity resources with minimal energy consumption, especially when multiple choices are available for each request ( $\mathcal{N} \geq 13$ ). Similarly, regardless of whether all requests can be supported (C and D, with  $\mathcal{R} \leq 200$ ), near-optimal solutions are consistently attained. However, D3QL exhibits less efficiency and stability compared to ORIENT, mainly due to its inferior state decoding capability.

## VI. CONCLUSION

In this paper, we examined the joint problem of service instance placement and assignment, path selection, and request prioritization, dubbed PIRA, with the objective of maximizing the overall profit of the system (as a function of the number of supported concurrent requests) while minimizing the overall energy consumption over a continuous period of time, taking into account E2E latency and resource capacity constraints. This problem was formulated as a MINLP problem, its complexity was analyzed, and it was demonstrated that it is an NP-hard problem. Subsequently, a technique named ORIENT was introduced to address the problem in a near-optimal manner, utilizing a GNN-empowered D3QL strategy. The effectiveness of the suggested technique was validated through numerical results. As potential future work, our intention is to tackle the problem in the context of dynamic environments characterized by temporal/spatial fluctuations in requests and resources.

## ACKNOWLEDGMENT

It is partially supported by the European Union’s Horizon 2020 Research and Innovation Program through the

aerOS project under Grant No. 101069732; the Business Finland 6Bridge 6Core project under Grant No. 8410/31/2022; the European Union’s HE research and innovation program HORIZON-JUSNS-2022 under the 6GSandbox project (Grant No. 101096328); and the Research Council of Finland 6G Flagship Programme under Grant No. 346208. This research was also conducted at ICTFICIAL Oy. The paper reflects only the authors’ views, and the commission bears no responsibility for any utilization of the information contained herein.

## REFERENCES

- [1] M. Latva-aho, “Key Drivers and Research Challenges for 6G Ubiquitous Wireless Intelligence,” 2019, publisher: University of Oulu.
- [2] H. Yu *et al.*, “Toward 6G-Based Metaverse: Supporting Highly-Dynamic Deterministic Multi-User Extended Reality Services,” *IEEE Network*, vol. 37, no. 4, pp. 30–38, 2023.
- [3] H. Mazandarani *et al.*, “A Semantic-Aware Multiple Access Scheme for Distributed, Dynamic 6G-Based Applications,” *arXiv preprint arXiv:2401.06308*, 2024.
- [4] J. Prados-Garzon *et al.*, “Deterministic 6GB-Assisted Quantum Networks with Slicing Support: A New 6GB Use Case,” *IEEE Network*, 2023.
- [5] Z. Yang *et al.*, “Increasing the Energy Efficiency of a Data Center Based on Machine Learning,” *Journal of Industrial Ecology*, vol. 26, no. 1, pp. 323–335, 2022.
- [6] “Cisco Annual Internet Report (2018–2023) White Paper.”
- [7] H. Xuan *et al.*, “Multi-Agent Deep Reinforcement Learning Algorithm with Self-adaption Division Strategy for VNF-SC Deployment in SDN/NFV-Enabled Networks,” *Applied Soft Computing*, vol. 138, p. 110189, May 2023.
- [8] R. Solozabal *et al.*, “Virtual Network Function Placement Optimization With Deep Reinforcement Learning,” *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 2, pp. 292–303, Feb. 2020.
- [9] C. Pham *et al.*, “Traffic-Aware and Energy-Efficient VNF Placement for Service Chaining: Joint Sampling and Matching Approach,” *IEEE Trans. on Services Computing*, vol. 13, no. 1, pp. 172–185, Jan. 2020.
- [10] G. L. Santos *et al.*, “Availability-Aware and Energy-Aware Dynamic SFC Placement using Reinforcement Learning,” *The Journal of Supercomputing*, vol. 77, no. 11, Nov. 2021.
- [11] Z. Sasan *et al.*, “Joint Network Slicing, Routing, and In-Network Computing for Energy-Efficient 6G,” *arXiv preprint arXiv:2401.06306*, 2024.
- [12] M. Farhoudi *et al.*, “Qos-Aware Service Prediction and Orchestration in Cloud-Network Integrated Beyond 5G,” *arXiv preprint arXiv:2309.10185*, 2023.
- [13] N. He *et al.*, “Leveraging Deep Reinforcement Learning With Attention Mechanism for Virtual Network Function Placement and Routing,” *IEEE Trans. on Parallel and Distributed Systems*, vol. 34, no. 4, Apr. 2023.
- [14] M. Shokrnezhad *et al.*, “Double Deep Q-Learning-based Path Selection and Service Placement for Latency-Sensitive Beyond 5G Applications,” *IEEE Transactions on Mobile Computing*, pp. 1–14, 2023.
- [15] Z. Wang *et al.*, “Dueling Network Architectures for Deep Reinforcement Learning,” in *Proceedings of The 33rd International Conference on Machine Learning*, vol. 48, Jun. 2016, pp. 1995–2003.
- [16] V. Mnih *et al.*, “Human-level Control through Deep Reinforcement Learning,” *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.