Proximal Policy Optimization (PPO)-based Task Offloading in 3D Edge Computing

Luiz G. S. C. S. Segundo*, Renata dos Reis*, Andson Balieiro*, Tarik Taleb†, Dalton C. Valadares‡
*Centro de Informática (CIn), Universidade Federal de Pernambuco (UFPE), Recife, Brazil
†Faculty of Electrical Engineering and Information Technology, Ruhr University Bochum, Bochum, Germany
‡Colégio Agrícola Vidal de Negreiros (CAVN), Universidade Federal da Paraíba (UFPB), Bananeiras, Brazil
{lgscss, rkgr, afc9, amb4}@cin.ufpe.br, tarik.taleb@ruhr-uni-bochum.de, dalton.valadares@embedded.ufcg.edu.br

***Abstract*—Computational offloading in three-dimensional (3D) edge computing networks enables simultaneous use of terrestrial and aerial platforms. Selecting optimal nodes and allocating computing and transmission resources for task offloading is NP-hard, especially when considering mechanisms such as task partitioning, replication, and parallel offloading to improve latency and reliability. This paper proposes a Proximal Policy Optimization (PPO)-based solution that jointly optimizes task partitioning and replication, node selection, and resource allocation, while accounting for potential failures in computation, bidirectional transmissions, and result fusion at the user device. Results show that our solution achieves consistent performance across varying task dimensions, partition counts, and replica limits, effectively balancing latency, reliability, efficiency, and energy consumption.**

***Index Terms*—3D Edge Computing, Task Offloading, Resource Allocation, Proximal Policy Optimization.**

I. Introduction

Three-Dimensional (3D) Multi-Access Edge Computing (MEC) integrates terrestrial and aerial computing resources located near users, reducing latency and core network load [1], and enabling latency-critical and computation-intensive applications in 5G and upcoming 6G networks. Unlike traditional MEC, which often depends on enterprise-based terrestrial infrastructures and may encounter limitations in remote, dense, or disrupted scenarios, 3D MEC incorporates idle resources from Extreme Edge Computing (XEC), beyond the Radio Access Network (RAN), such as wearables and connected vehicles [2], as well as aerial platforms like Unmanned Aerial Vehicles (UAVs), to expand processing capacity and enhance flexibility.

3D MEC enables computational offloading, allowing users to process tasks remotely to conserve local energy or make task execution feasible through both horizontal and vertical offloading. Partitioning a task allows idle nearby resources, including low-power XEC devices, to process part of the workload so that no single resource-constrained device must execute it entirely, while parallelism and replication can reduce latency and improve reliability at the cost of additional communication and energy. However, the joint optimization of node selection and resource allocation remains NP-hard [3], especially when considering practical aspects such as bidirectional communication, computation and transmission failures, result fusion at the user equipment (UE), and advanced mechanisms including task partitioning, replication, and parallel offloading.

Several studies have investigated task offloading in UAV-enabled MEC systems using mathematical optimization and reinforcement learning [1], [4]–[11]. Recent PPO-based works focus mainly on UAV deployment, positioning, trajectory, and computation offloading [9]–[11], but most studies neglect transmission and computation failures, downlink communications, result fusion at the UE, or the joint optimization of partitioning, replication, node selection, and resource allocation, limiting their applicability to realistic 3D scenarios. Evolutionary techniques such as Genetic Algorithms (GA) are also widely used for such NP-hard problems, but typically require a new search for each problem instance and do not yield a reusable policy, which may limit their use in dynamic edge environments.

To address these limitations, this paper extends our previous GA-based work [12], retaining its 3D edge system model and optimization framework while introducing a PPO-based solution that jointly optimizes subtask partitioning and replication, node selection, and resource allocation, explicitly accounting for computation and bidirectional transmission failures and result fusion at the UE. The main contributions are: (i) the reformulation of the problem as a hybrid discrete–continuous reinforcement learning task, (ii) a multi-head PPO actor–critic adapted to the heterogeneous offloading decision space, (iii) the analysis of penalty and non-penalty reward formulations, and (iv) a direct comparison against the GA-based approach in [12], Random, and Ateya's heuristic. The system model and evaluation settings follow [12] to enable direct comparison. Results show an effective trade-off among latency, reliability, energy consumption, and efficiency across task dimensions, partition levels, and replication configurations.

II. System Model

The system comprises U users, u_i. At time t, each user u_i has a task $T_i(t)$, with $T_i \in \mathbb{T}_t$, which involves processing $D_i(t)$ bits, requiring $Q_i(t)$ CPU cycles per bit. The 3D MEC provides computation and connectivity resources for users to process their tasks via nodes with varying computational capabilities, communication features, and reliability. In addition to the user equipment (UE) resources, the system comprises M terrestrial MEC nodes, V UAVs, and E XEC nodes, as illustrated in Fig. 1. From the user perspective, $\mathbb{K} = \{k_j\}$ denotes the set of available nodes that can be used to process tasks, with $j = 0, 1, ..., M+V+E$. Here, 0 indicates UE; 1 to M represent the terrestrial MEC nodes; $(M+1)$ to $(M+V)$ refers the UAVs, and $(M+V+1)$ to $(M+V+E)$ denotes the extreme nodes. Each node k_j has the CPU clock frequency, capacitance, and reliability level denoted as f_j, z_j, and r_j,

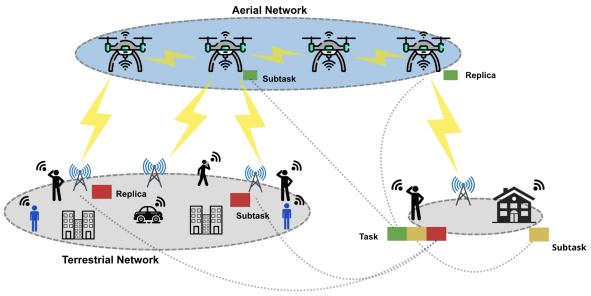


Fig. 1. Task partitioning, replication, and assignment in the 3D MEC scenario.

respectively.

Moreover, since the user devices and the XEC nodes may concurrently run other applications from their owners, their available resources to process a task $T_i(t)$ may fluctuate over time t, denoted by $\theta_j(t)$, with $j = 0, M+V+1, ..., M+V+E$. For the remaining remote nodes, we assume full resource availability, i.e., $\theta_j(t) = 1$, with $k_j \in (\mathbb{K} - \{k_0\} - \{k_{M+V+1}, ..., k_{M+V+E}\})$.

We consider a task $T_i(t)$ can be partitioned into subtasks, which may be replicated and processed across multiple nodes. This strategy can improve latency and reliability but at the cost of higher energy consumption and resource utilization. The results of subtasks processed on remote nodes are transmitted back to the UE for consolidation, completing the task execution. Thus, our study accounts for bidirectional transmissions, uplink and downlink, as well as potential failures during processing and communication, which may affect the overall task reliability. In addition, the remote nodes are assumed to operate on orthogonal frequencies [13] to mitigate interference arising from simultaneous transmissions.

A centralized controller [4] gathers task, device, node, and channel information, decides the task-to-node mapping and resource allocation, and informs the UEs.

A. Computing Model

Given a task $T_i(t)$ divided into n_i disjoint subtasks $s_{ic}(t)$, each subtask has size $d_{ic}(t) = \phi_{ic}(t)D_i(t)$ bits, with $c = 1, 2, \ldots, n_i$, $n_i \in \mathbb{N}$, $\phi_{ic}(t) \in \mathbb{R}$, $0 < \phi_{ic}(t) \leq 1$, and $\sum_{c=1}^{n_i} \phi_{ic}(t) = 1$. Some subtasks may be replicated to improve reliability, where $\mathbb{S}_{ic}$ denotes the set of copies of subtask $s_{ic}(t)$ and $q_{ic} = |\mathbb{S}_{ic}|$, with $\sum_c q_{ic} \leq |\mathbb{K}|$.

Let $\mathbb{K}_i(t) \subseteq \mathbb{K}$ denote the set of nodes selected by user u_i at time t to process its subtasks. The latency to process subtask s_{ic} or one of its replicas when assigned to node $k_j \in \mathbb{K}_i(t)$ is given by Eq. 1. The binary variable $o_{icj}(t)$ assumes value 1 when node k_j is assigned to process subtask s_{ic} of user u_i at time t. Finally, $\alpha_{ij}(t) \in \mathbb{R}^+$, with $|\alpha_{ij}(t)| \leq 1$ and $\sum_i \alpha_{ij}(t) \leq 1$, denotes the CPU fraction of node k_j allocated to this processing.

The results of subtasks processed on remote nodes are consolidated at the user side (k_0), which incurs additional delay and energy consumption. Let $\delta_i(t)$, $0 < \delta_i(t) \leq 1$, denote the ratio between the amount of data offloaded to remote nodes and that returned to the user device for fusion. The data fusion latency of task T_i at time t is given by $L_{f_i}(t)$, where the summation considers only remote nodes. Similar to $D_i(t)$ and $Q_i(t)$, the value of $\delta_i(t)$ depends on the application characteristics and can be determined through application profiling.

$$Lp_{ic}(t) = [d_{ic}(t)Q_i(t)o_{icj}(t)]/[\alpha_{ij}(t)\theta_j(t)f_j] \quad (1)$$

$$L_{f_i}(t) = \frac{Q_i(t)\delta_i(t)}{\theta_0(t)f_0} \sum_{c=1}^{n_i} d_{ic}(t)o_{icj}(t) \quad (2)$$

The energy in Joules (J) spent to process the subtask s_{ic} on the node k_j is given by Eq. 3, and the total energy consumption for processing task T_i is denoted by Eq. 4, which considers all subtask copies, adopts Eq. 3 to compute $Ep_{ic'}(t)$, and includes the energy consumed for data fusion.

$$Ep_{ic}(t) = z_j[\alpha_{ij}(t)\theta_j(t)f_j]^3 Lp_{ic}(t) \quad (3)$$

$$Ep_i(t) = \sum_{c=1}^{n_i} \sum_{c' \in \mathbb{S}_{ic}} Ep_{ic'}(t) + z_0[\alpha_{i0}(t)\theta_0(t)f_0]^3 L_{f_i}(t) \quad (4)$$

Since task processing may fail due to hardware or software errors, we consider the node reliability r_j, defined as the complement of the failure probability of each computing node k_j. It can be estimated using the mean time between failures and the mean time to repair, derived from historical failure and maintenance records [14]. Given that task $T_i(t)$ may be processed across multiple nodes with admitted redundancy, and letting $\mathbb{S}_{ic}$ denote the set of copies of subtask s_{ic}, the overall processing reliability of T_i is determined by Eq. 5

$$Rp_i(t) = \prod_{c=1}^{n_i} \left[1 - \prod_{c' \in \mathbb{S}_{ic}} (1 - r_j o_{ic'j}(t))\right] \quad (5)$$

B. Communication Model

We assume a channel quality indicator (CQI)-based adaptive modulation and coding scheme (MCS) for both transmission directions. The CQI is periodically reported from the UE to the base station, allowing dynamic adjustment of the modulation and coding schemes to maintain the block error rate (BLER) below a predefined threshold. We follow the 3GPP TS 38.214 specification to determine the spectral efficiency (SE) from a given CQI and indexing the MCS table, as in [15].

Given that the subtask $s_{ic}(t)$, $c = 2, 3, ..., n_i$ is offloaded to the remote node k_j, the uplink and downlink transmission latencies, $L_{up_{icj}}(t)$ and $L_{down_{icj}}(t)$, are given by Eq. 6, where d_{ic} is the subtask size in bits and δ_i denotes the ratio of data returned to the UE. $V_{up_{ij}(t)}$ and $V_{down_{ij}(t)}$ are the uplink and downlink data rates between u_i and k_j, given by Eq. 7, where the transmission block size (TBS) follows 3GPP TS 38.214 Release 16 [15], and OH is the PDSCH/PUSCH overhead. The number of RBs for downlink or uplink is given by Eq. 8, where $\beta_{ij}(t) \in [0, 1]$ is the fractional bandwidth allocated between u_i and k_j, and B is the total bandwidth.

$$L_{up_{ic}}(t) = \frac{d_{ic}(t)}{V_{up_{ij}(t)}} o_{icj}(t); L_{down_{ic}}(t) = \frac{\delta_c(t)d_{ic}(t)}{V_{down_{ij}}(t)} o_{icj}(t) \quad (6)$$

$$V_{down,up_{ij}}(t) = \frac{bits_{slot}}{slot_{duration}} = \frac{(1-OH) * TBS}{(1/2^{\mu})10^{-3}} \quad (7)$$

$$n_{PRB_{ij}} = \lceil (\beta_{icj}(t)B)/(2^{\mu}180) \rceil \quad (8)$$

Considering that subtasks $s_{ic}(t)$ are replicated into $q_{ic}(t)$ copies, the latency $L_{ic}(t)$, which includes the uplink and downlink transmissions, and processing time, is given by the minimum individual latency $L_{ic'}(t)$ among the replicas $s_{ic'}(t) \in \mathbb{S}ic$, as shown in Eq. 9. Here, $Lic'(t)$ is computed using Eq. 10, where for a copy $s_{ic'}(t)$ processed on user device k_0, the components $L_{up_{ic'}}(t)$ and $L_{down_{ic'}}(t)$ are zero. Consequently, the total latency for completing task T_i, accounting the data fusion delay, is given by Eq. 11. The transmission reliability refers to the successful transfer of data packets over a communication link. The transmission failure probability Pf_{ic} of each subtask c is defined by Eq. 14, where $Pf_{ic'_{up}}(t)$ and $Pf_{ic'_{dw}}(t)$ represent the failure probabilities for uplink and downlink transmissions of the copy c', respectively. TBS_{up} and TBS_{dw} are the transport block sizes adopted in uplink and downlink directions, as in [15]. The transmission reliability of T_i at time t, $Rt_i(t)$, and its overall reliability are given by Eq. 15 and Eq. 16, respectively. For reliability purpose, subtasks (replicas) of the same task are not allowed to be processed on the same node.

$$L_{ic}(t) = \min_{s_{ic'}(t) \in \mathbb{S}_{ic}} \{L_{ic'}\} \quad (9)$$

$$L_{ic'}(t) = L_{up_{ic'}}(t) + L_{down_{ic'}}(t) + Lp_{ic'}(t) \quad (10)$$

$$L_i(t) = L_{f_i}(t) + \max_c \{L_{ic}(t)\} \quad (11)$$

$$Pf_{ic'_{up}}(t) = 1 - (1 - BLER)^{\frac{d_{ic}(t) o_{ic'j}(t)}{TBS_{up}}} \quad (12)$$

$$Pf_{ic'_{dw}}(t) = 1 - (1 - BLER)^{\frac{d_{ic}(t) * \delta_i * o_{ic'j}(t)}{TBS_{dw}}} \quad (13)$$

$$Pf_{ic}(t) = \prod_{c' \in \mathbb{S}_{ic}} \left[Pf_{ic'_{up}}(t) + Pf_{ic'_{dw}}(t) - Pf_{ic'_{up}}(t) Pf_{ic'_{dw}}(t) \right] \quad (14)$$

$$Rt_i(t) = \prod_{c=1}^{n} (1 - Pf_{ic}(t)) \quad (15)$$

$$R_i(t) = Rp_i(t) Rt_i(t) \quad (16)$$

The energy consumption for uplink and downlink of T_i considers the UE (W_i) and remote node (W_j) transmission powers, and latencies for each subtask c and its copies c', $L_{down_{ic'}}(t)$ and $L_{up_{ic'}}(t)$, as seen in Eqs. 18 and 17. The total energy consumption for offloading task T_i is given by Eq. 19, which accounts for the energy spent on uplink and downlink transmissions, task processing, and result fusion.

$$E_{up_i}(t) = \sum_{c=1}^{n_i} \sum_{c' \in \mathbb{S}_{ic}} W_i * L_{up_{ic'}}(t) \quad (17)$$

$$E_{down_i}(t) = \sum_{c=1}^{n_i} \sum_{c' \in \mathbb{S}_{ic}} W_j * L_{down_{ic'}}(t) \quad (18)$$

$$E_i(t) = E_{down_i}(t) + E_{up_i}(t) + E_{p_i}(t) \quad (19)$$

C. Problem Formulation

Deciding where to process tasks and how many resources to allocate affects task latency, reliability, and energy consumption. The task offloading optimization problem is thus formulated as follows: given a set of tasks at each instant t, a set of nodes with varying computing and communication capacities, and allowing task splitting and subtask replication, determine the optimal task partitioning, subtask replication, subtask-node assignments, and resource (CPU and bandwidth) allocation to minimize the average energy consumption ($\overline{E}(t)$) and latency ($\overline{L}(t)$) while maximizing the average reliability ($\overline{R}(t)$):

$$\min \ \overline{E}(t), \ \overline{L}(t), \quad \max \ \overline{R}(t) \quad (20)$$

$$\text{s.t.} \ \sum_c \phi_{ic}(t) = 1, \forall \ T_i \quad (21)$$

$$\sum_c o_{icj}(t) \leq 1, \forall \ (T_i, k_j) \quad (22)$$

$$\sum_i \alpha_{ij}(t) \leq 1, \forall \ k_j, \quad \sum_i \beta_{ij}(t) \leq 1, \forall \ k_j \quad (23)$$

$$0 < \phi_{ic}(t), \alpha_{ij}(t), \beta_{ij}(t) \leq 1 \in \mathbb{R}, \quad o_{icj}(t) \in \{0, 1\} \quad (24)$$

III. PPO-Based Solution

A Proximal Policy Optimization (PPO)-based solution is proposed to address the optimization problem in Eq. 20. PPO is adopted due to its stable policy updates through the clipped surrogate objective [16], and because its stochastic policy accommodates both categorical and Gaussian heads for the hybrid discrete–continuous action space of the task offloading problem.

Since the optimization problem involves both discrete and continuous decisions, including task partitioning, replication, node selection, and resource allocation, we adopt a Multi-Head Actor–Critic architecture. Each Actor head models one decision component, while the Critic estimates the state-value function using Generalized Advantage Estimation (GAE). Policy parameters are optimized using the standard PPO clipped objective:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t \Big[\min \Big(r_t(\theta) \hat{A}_t, \ \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A}_t \Big) + c_{ent} \mathcal{S}[\pi_\theta] \Big] \quad (25)$$

where $r_t(\theta)$ denotes the probability ratio between the updated and previous policies, $\hat{A}_t$ represents the advantage estimate, ϵ is the clipping parameter, and $\mathcal{S}[\pi_\theta]$ is the entropy of the policy, encouraging sufficient exploration.

A. State and Action Spaces

At each decision step t, the centralized controller observes the environment through a state vector $s_t \in \mathbb{R}^{|s|}$ constructed by concatenating four per-node resource arrays and two scalar task-tracking variables:

$$s_t = \left[\alpha_j^{\text{av}}, \ \beta_j^{\text{up,av}}, \ \beta_j^{\text{dw,av}}, \ \nu_j, \ \mathcal{P}(t), \ u(t) \right] \quad (26)$$

where $j = 0, 1, \ldots, |\mathbb{K}| - 1$, with $|\mathbb{K}| = U + M + V + E = 60$ total nodes; $\alpha_j^{\text{av}} \in [0, 1]$ is the available CPU fraction of node k_j; $\beta_j^{\text{up,av}}$, $\beta_j^{\text{dw,av}} \in [0, 1]$ are the available uplink and downlink

bandwidth fractions; $\nu_j \in \{0,1\}$ indicates whether k_j is active; $\mathcal{P}(t)$ is the count of tasks processed so far in the episode; and $u(t)$ is the index of the UE currently being served. The resulting state dimension is $|s_t| = 4\,|\mathbb{K}| + 2 = 242$. All components are normalised to $[0,1]$ using running mean-and-variance statistics before being fed to the networks.

The action vector $a_t \in \mathbb{R}^{|a|}$ encodes all decisions required to offload one task T_i:

$$a_t = \big[\ \underbrace{\phi_{ic}}_{\text{subtask}},\ \underbrace{\beta^{\text{up}}_{ic'},\ \beta^{\text{dw}}_{ic'},\ \alpha_{ic'}}_{\text{resources}},\ \underbrace{\tilde{q}_{ic}}_{\text{replicas}},\ \underbrace{\tilde{o}_j}_{\text{offload}}\ \big] \tag{27}$$

The multi-head design reflects the heterogeneous structure of this action space: a shared state representation captures common system information, while specialized Actor heads model decision components with different action characteristics.
Let $b = n + n \cdot q_{\max} = 20$ denote the block size covering both primary subtasks and their replicas. The six output heads of the Actor are described as follows. *Subtask partitioning* ϕ_{ic} ($n = 4$ values): a categorical head outputs n logits; the selected index is converted to a one-hot encoding and normalised via softmax to satisfy $\sum_c \phi_{ic} = 1$. *Resource allocation* $\beta^{\text{up}}_{ic'}$, $\beta^{\text{dw}}_{ic'}$, $\alpha_{ic'}$ ($b = 20$ values each): three Gaussian heads output means bounded by $\tanh$, with learnable log-standard-deviations $\log\sigma$ initialised at -2; raw samples are clipped to $[-10, 10]$ and mapped to $[0,1]$ via Sigmoid before being applied to the environment. *Replication level* $\tilde{q}_{ic}$ ($n = 4$ values): a Gaussian head produces a continuous score in $[0,1]$ after Sigmoid; the discrete replica count is recovered as

$$q_{ic} = \text{round}(q_{\max} \cdot \tilde{q}_{ic})\,, \quad q_{ic} \in \{0, 1, \ldots, q_{\max}\} \tag{28}$$

with $q_{\max} = 4$. *Node selection* $\tilde{o}_j$ ($|\mathbb{K}| = 60$ values): a Gaussian head outputs one continuous score per node after Sigmoid; the $b = 20$ nodes with the highest scores are selected and assigned in descending order, with the first $n = 4$ to the principal subtasks and the remaining $n \cdot q_{\max} = 16$ to their replicas. Node IDs map to types as: 0–9 (UE), 10–24 (MEC), 25–39 (UAV), and 40–59 (XEC). The total action dimension is $|a_t| = n + 3b + n + |\mathbb{K}| = 128$. Unless stated otherwise, $n = q_{\max} = 4$ (Table I).

The continuous-to-discrete mappings for replication and node selection retain continuous policy outputs during learning while producing feasible discrete decisions for the environment. Since the state dimension grows as $4|\mathbb{K}| + 2$ and the action dimension depends linearly on $|\mathbb{K}|$ and $b = n(1 + q_{\max})$, the representation sizes grow linearly with the number of nodes and the number of subtask copies considered. The present evaluation considers 60 nodes; scalability to substantially larger topologies remains subject to future evaluation.

B. Reward Function

The reward function guides the agent towards maximizing system utility by jointly minimizing task latency (L_i) and energy consumption (E_i), while maximizing reliability (R_i). To analyze the impact of failure handling during learning, we investigate two reward formulations. The penalty formulation (Eq. 29) introduces a gradual penalty: when allocation fails, the agent receives a negative reward proportional to the fraction of subtasks and replicas not successfully allocated; otherwise, the reward follows the non-penalty case, described next. The non-penalty formulation (Eq. 30) assigns a positive reward, proportional to reliability and inversely proportional to latency and energy, only when a task is successfully allocated; when allocation fails, no penalty is applied, resulting in a sparse reward signal. E_m is a fixed energy normalization factor.

$$R_P = \begin{cases} \dfrac{E_m \times R_i}{E_i \times L_i + 1}, & \text{if } T_i(t) \text{ is allocated} \\ -\dfrac{s^{\text{fail}}_{ic} + q^{\text{fail}}_i}{n_i + q_i}, & \text{if } T_i(t) \text{ fails} \end{cases} \tag{29}$$

$$R_{NP} = \begin{cases} \dfrac{E_m \times R_i}{E_i \times L_i + 1}, & \text{if } T_i(t) \text{ is allocated} \\ 0, & \text{if } T_i(t) \text{ fails} \end{cases} \tag{30}$$

The reward explicitly captures the trade-off among the objectives: lower latency and energy increase the reward, whereas higher reliability is favored. Since replication and parallelism may improve reliability or latency at the cost of additional energy consumption and communication overhead, the agent learns a compromise among these competing objectives.

IV. Result Analysis

We evaluate our PPO-based solution in terms of latency, reliability, energy consumption, and efficiency (bits processed/transmitted per joule) across three scenarios: task dimensionality (Section IV-B), varying task size (D_i) and complexity (Q_i) as low (L), moderate (M), and high (H) [1]; task partitioning levels (Section IV-C), assessing the effect of parallelism; and the maximum number of allowed subtask replicas (Section IV-D). We compare our solution against three schemes: Random, where all decision variables are chosen randomly and tasks are mapped in parallel; Ateya, which follows [17] by mapping tasks sequentially to a single node without partitioning or replication, prioritizing latency and then energy consumption; and a Genetic Algorithm (GA)-based scheme [12], which encodes task partitioning, subtask replication, node selection, and resource allocation as a chromosome and evolves solutions through genetic operators.

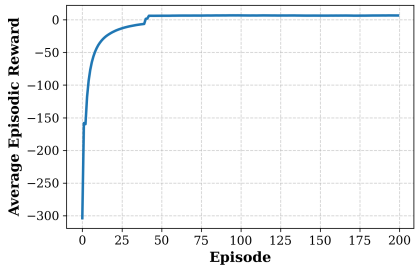
The Actor uses a shared fully connected layer (256 neurons, ReLU) followed by the six output heads described in Section III-A, while the Critic uses two fully connected layers (128 neurons each, ReLU) followed by a linear output for $V(s)$. Hyperparameters were selected based on empirical tuning and prior literature [16]: $\gamma = 0.99$, $\lambda = 0.95$, $\epsilon = 0.2$, and $c_{ent} = 0.005$. Actor and Critic were trained with Adam using learning rates 3×10^{-4} and 1×10^{-3}, respectively, batch size 64, and four epochs per policy update.

A. PPO Convergence

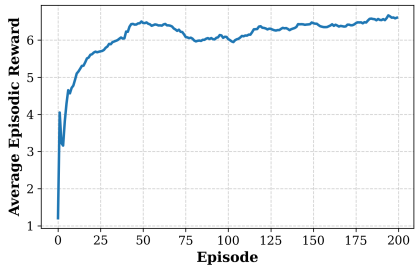
Prior to evaluating the performance metrics, we analyze PPO convergence through the cumulative reward averaged across independent runs. As shown in Fig. 2, both reward formulations exhibit increasing rewards followed by stable behavior, indicating convergence under the adopted multi-head architecture and hyperparameter configuration.

TABLE I
SIMULATION PARAMETERS

Scenario		**D (MB)**	**Q(cycle/bit**	**n**	**q**
Dimension	L	(0.6,2)	(60, 100)	4	4
	M	(5,10)	(120, 240)	4	4
	H	(20,60)	(120, 240)	4	4
Parallelism		(0.6, 2)	(60, 100)	2,4,6	4
Replication		(0.6, 2)	(60, 100)	4	2,4,6
Parameters		User	Ext.	Uav	Mec
# Nodes		10	20	15	15
f_j (GHz)		(1,2)	(1,2)	(1,3)	(20,25)
z_j		10^{-27}	10^{-27}	10^{-28}	10^{-28}
r_j		(0.9999, 0.99999)		(0.9955, 0.9999)	
Wi, Wj (W)		(0.08,0.1)	(0.08,0.1)	(0.18, 0.22)	(1,2)
$\theta_j(t)$		(0.7,1)	(0.7, 1)	1	1
$SINR$ (dB)		(10,25)			
$nPRB$		100 (Uplink), 100 (Downlink)			
$BLER$, μ, δ_i		0.00001, 3, (0.001, 0.4)			



(a) Penalty-based.



(b) Non-penalty.

Fig. 2. Reward convergence of PPO using different reward functions.

B. Task Dimension Scenario (D_i, Q_i)

This section evaluates the impact of task size D_i and complexity Q_i across schemes. As shown in Fig. 3(a), both PPO variants achieve lower latency than Random across all dimensionalities, while Ateya, which avoids partitioning and data fusion, maintains the lowest latency overall. However, this superior latency comes at the cost of significantly higher energy consumption and reduced reliability, as shown in Fig. 3(b) and 3(c). Compared with the proposed PPO solutions, the Ateya scheme consumes approximately 70× more energy while reducing reliability by up to 2.20% in the moderate scenario.

The PPO approach exhibits an almost consistent behavior across increasing task dimensionalities, attaining the lowest energy consumption and high reliability values (above 0.90 in all cases) for both reward formulations. It also achieves the highest efficiency among all schemes in the moderate and high cases (Fig. 3(d)). Random, despite underperforming PPO, leverages parallelism and redundancy to outperform Ateya in reliability and energy (gains up to 0.05 and 62%, respectively, for high-dimensionality tasks). The GA baseline achieves latencies of 0.66s, 6.89s, and 28.34s (low, moderate, high) and outperforms Ateya in energy by 85%, benefiting from partitioning and replication. Still, PPO remains more energy-efficient: Penalty PPO cuts energy by 73% versus GA in the low case with comparable reliability (0.99).

C. Task Parallelism Scenario (n)

Increasing the number of partitions does not lower PPO's latency (Fig. 4(a)): the additional simultaneous transmissions and fusion work offset the gain from parallelism and add failure points that can affect reliability. Similarly to the previous scenario, Ateya achieves lower latency but at a significant energy cost (Fig. 4(b)). Non-Penalty PPO instead balances competitive latency with high reliability (0.91–0.92) and much lower energy (96.8–98.3% below Ateya across all n), with efficiency rising 22.25% and energy growing only 13.10% from $n = 2$ to $n = 6$ despite an intermediate jump at $n = 4$ (Figs. 4(b),(d)).

Penalty PPO's energy and latency also rise with more partitions, a diminishing return caused by the added cost of simultaneous transmissions offsetting the benefit of selecting energy-efficient nodes. GA outperforms Ateya in energy across all n by jointly exploring replication and node selection, reaching 0.99 reliability at $n = 6$; yet PPO still outperforms GA in energy consumption, with Non-Penalty PPO cutting energy by 83.3% relative to GA at $n = 6$ at slightly lower reliability.

D. Task Replication Scenario (q_i)

Reliability and efficiency of Non-Penalty PPO vary with the maximum replicas q, peaking at $q = 4$ (Figs. 5(c),(d)), because the agent trades off redundancy against energy rather than maximizing replication outright. In contrast, Penalty PPO's reliability instead stays high and nearly constant across all q, ensuring stability. Both PPO strategies keep markedly lower energy than the baselines with controlled variation between $q = 2$ and $q = 6$ (Fig. 5(b)), strategically allocating replicas to energy-efficient nodes while preserving reliability; at $q = 6$, PPO cuts energy by 95.65% relative to Ateya, which stays nearly constant across q (no replication).

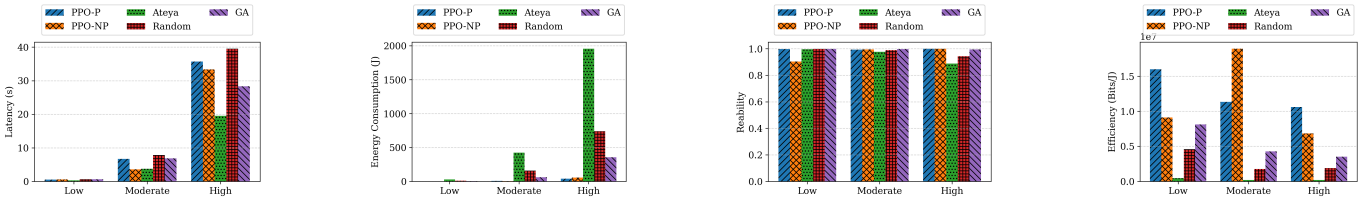
GA also reaches high reliability with more replicas (0.99 at $q = 6$, close to Random), suggesting reliability converges across schemes given sufficient replication. PPO remains more energy-efficient: Penalty PPO cuts energy by 70–73% versus GA at $q = 6$ while achieving comparable reliability, indicating a more favorable energy–reliability trade-off in this scenario.

E. Execution Time

We also compare the execution time needed to map 10 tasks. GA must run a new search for every problem instance and takes 52.7–109.1s across all scenarios, whereas the trained PPO policy takes 0.47–1.03s (Non-Penalty) and 0.64–2.23s (Penalty), i.e., roughly 35–113× faster than GA, without per-instance optimization.

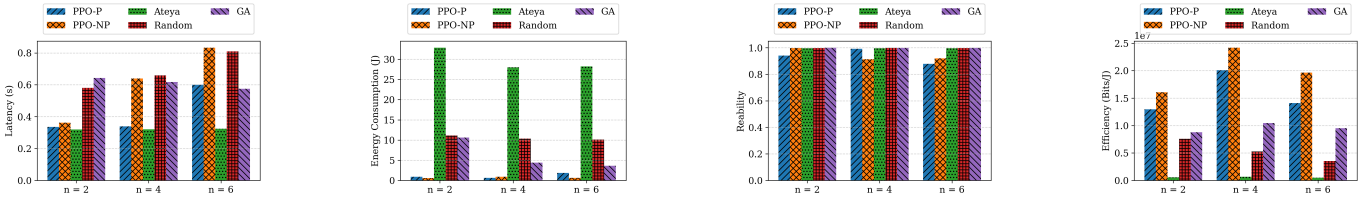
V. CONCLUSION

This work addressed task offloading in 3D Edge Computing Networks through a PPO-based solution jointly considering task partitioning and replication, node selection, resource allocation, computation and transmission failures, and data fusion at the UE. The proposed approach achieved a favorable trade-off



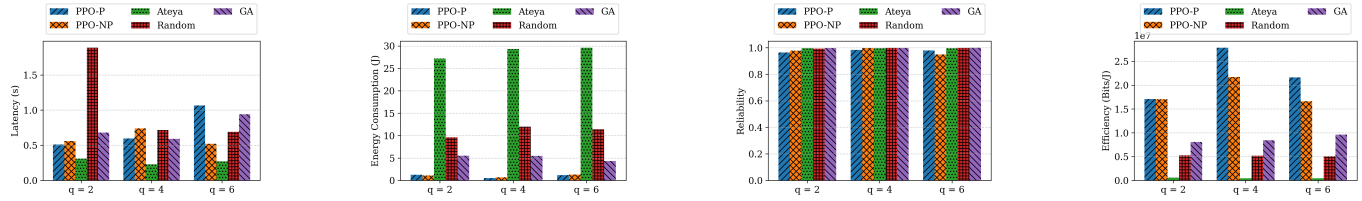
(a) Latency. (b) Energy Consumption. (c) Reliability. (d) Efficiency.

Fig. 3. Results varying the task dimensionality.



(a) Latency. (b) Energy Consumption. (c) Reliability. (d) Efficiency.

Fig. 4. Results varying the number of partitions for tasks (parallelism level).



(a) Latency. (b) Energy Consumption. (c) Reliability. (d) Efficiency.

Fig. 5. Results varying the number of replicas for tasks (replication level).

among latency, reliability, and energy consumption across different task complexities, parallelism levels, and replication settings.

PPO reduced energy consumption by up to 83.3% compared with GA with only a slight reduction in reliability, and by up to 95% compared with Ateya in high-redundancy scenarios. Moreover, the trained PPO policy was roughly 35–113$\times$ faster than GA for mapping 10 tasks, avoiding per-instance optimization. Future work will investigate generalization to unseen configurations, scalability to larger networks, and PPO training cost.

References

[1] I. A. Elgendy, S. Meshoul, and M. Hammad, "Joint task offloading, resource allocation, and load-balancing optimization in multi-uav-aided mec systems," *Applied Sciences*, vol. 13, no. 4, 2023.

[2] Y. Nabil, M. Abdelhadi, S. Sorour, H. ElSawy, S. A. Elsayed, and H. S. Hassanein, "Spatiotemporal analysis of parallelized computing at the extreme edge," *IEEE Transactions on Mobile Computing*, pp. 1–17, 2026.

[3] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Transactions on Networking*, vol. 24, no. 5, pp. 2795–2808, 2016.

[4] E. E. Haber, H. A. Alameddine, C. Assi, and S. Sharafeddine, "A reliability-aware computation offloading solution via uav-mounted cloudlets," in *IEEE Intl. Conf. on Cloud Networking (CloudNet)*, 2019.

[5] W. Shi, L. Chen, and X. Zhu, "Task offloading decision-making algorithm for vehicular edge computing: A deep-reinforcement-learning-based approach," *Sensors*, vol. 23, no. 17, 2023.

[6] H. Yu and H. Fei, "Task offloading optimization algorithm for uav assisted mobile edge computing network," in *9th International Conference on Communication, Image and Signal Processing*, 2024, pp. 123–128.

[7] J. Zhang, G. Zhang, X. Wang, X. Zhao, P. Yuan, and H. Jin, "Uav-assisted task offloading in edge computing," *IEEE IoT Journal*, 2025.

[8] M. Ahmed, N. Fatima, S. Raza, H. Ali, A. Qayum, W. Ullah Khan, M. Sheraz, and T. Chee Chuah, "Optimizing resource allocation and task offloading in multi-uav mec networks," *IEEE Access*, 2025.

[9] Y. Wang, X. Wu, J. Farooq, and J. Chen, "Coordinated UAV deployment and task offloading in UAV-assisted LEO satellite edge computing systems via proximal policy optimization," in *Proc. of the IEEE Intl. Conf. on Communications Workshops (ICC Workshops)*, 2025.

[10] Y. Wang, J. Farooq, H. Ghazzai, and G. Setti, "Joint positioning and computation offloading in multi-UAV MEC for low latency applications: A proximal policy optimization approach," *IEEE Transactions on Mobile Computing*, vol. 24, no. 10, pp. 9584–9598, 2025.

[11] L. Wu, J. Du, and J. Qu, "Joint trajectory, RIS, and computation offloading optimization via decentralized model-based PPO in urban multi-UAV mobile edge computing," *arXiv preprint arXiv:2603.20238*, 2026.

[12] R. d. Reis, C. de Souza, M. Falcão, A. Balieiro, and T. Taleb, "Task offloading in 3d edge computing networks," in *2026 IEEE Wireless Communications and Networking Conference (WCNC)*, 2026, pp. 1–6.

[13] J. Almutairi, M. Aldossary, H. A. Alharbi, B. A. Yosuf, and J. M. H. Elmirghani, "Delay-optimal task offloading for uav-enabled edge-cloud computing systems," *IEEE Access*, vol. 10, pp. 51 575–51 586, 2022.

[14] Z. Yang, C. Pan, K. Wang, and M. Shikh-Bahaei, "Energy efficient resource allocation in uav-enabled mobile edge computing networks," *IEEE Trans. on Wireless Comm.*, vol. 18, no. 9, pp. 4576–4589, 2019.

[15] P. Yoshioka, M. Damasceno, A. Balieiro, S. N. Swain, and E. Alves, "A decision-tree solution for the outdated cqi feedback problem in 5g networks," in *Brazilian Symp. on Comp. Systems Engineering*, 2024.

[16] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[17] A. A. Ashraf Ateya, A. Muthanna, R. Kirichek, M. Hammoudeh, and A. Koucheryavy, "Energy- and latency-aware hybrid offloading algorithm for uavs," *IEEE Access*, vol. 7, pp. 37 587–37 600, 2019.