Task Offloading in 3D Edge Computing Networks

Renata dos Reis *, Caio de Souza *, Marcos Falcão*, Andson Balieiro*, and Tarik Taleb†
*Centro de Informática (CIn), Universidade Federal de Pernambuco (UFPE), Recife, Brazil.
†Faculty of Electrical Engineering and Information Technology, Ruhr University Bochum, Bochum, Germany.
{rkgr, cbbs, mrmf, amb4}@cin.ufpe.br, tarik.taleb@ruhr-uni-bochum.de, caiobruno.souza@upe.br

***Abstract*—Computational offloading in three-dimensional (3D) edge computing networks occurs both horizontally and vertically, enabling simultaneous use of terrestrial and aerial platforms. Selecting optimal nodes and allocating computing and transmission resources for task offloading is NP-hard, particularly when considering task partitioning, replication, and parallel offloading to improve latency and reliability. This paper proposes an evolutionary 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. The proposed approach is evaluated in terms of latency, reliability, energy consumption, and efficiency under varying task dimensions, partition counts, and replica limits. Results show that our solution achieves better tradeoffs between latency, energy consumption, system robustness, and efficiency and outperforms baseline methods.**

***Index Terms*—3D Edge Computing, Task Offloading, Resource Allocation, Evolutionary Approach.**

I. Introduction

Multi-Access Edge Computing (MEC) enables latency-sensitive and compute-intensive applications on resource-constrained devices in 5G networks by bringing cloud resources closer to end users, thereby reducing latency and core network congestion [1]. However, traditional MEC deployments are typically restricted to enterprise-owned infrastructures, limiting its ability to deliver ultra-low latency at the farthest edge. To overcome this, Extreme Edge Computing (XEC) extends MEC paradigm by leveraging idle resources from devices beyond the Radio Access Network (RAN), such as wearables, sensors, and connected vehicles.

In this context, computational offloading becomes a key service, allowing users to delegate tasks to remote nodes and conserve energy. Nevertheless, terrestrial infrastructure alone may be insufficient in scenarios such as remote areas, large events, or natural disasters, due to connectivity limitations or deployment constraints. Thus, aerial elements such as Unmanned Aerial Vehicles (UAVs) can act as mobile MEC nodes, offering flexible deployment, improved line-of-sight connectivity, and reduced user energy consumption for data transmission [2]. These platforms, when integrated with terrestrial MEC, form three-dimensional (3D) edge computing networks [2] that enhance both coverage and computational capacity in 5G and future 6G architectures [2].

In 3D Edge Computing, offloading occurs horizontally and vertically, enabling simultaneous use of terrestrial and aerial nodes. However, selecting optimal nodes and allocating computing and transmission resources is an NP-hard problem [3], especially when considering advanced mechanisms such as task partitioning into subtasks, replication, and parallel offloading to enhance latency and reliability, and also result downlink and fusion at the user device.

Several solutions for task offloading in UAV-based MEC networks have been proposed. For instance, the work in [3] jointly optimizes UAV positions, task partitioning, user assignment, and resource allocation, incorporating redundancy. Other studies [1], [4], [5] address load balancing and energy–delay trade-offs, considering both communication and computation aspects. However, these works often simplify the environment by assuming perfect transmission conditions and negligible downlink traffic, while disregarding XEC elements and result fusion at the user device. Such assumptions are unrealistic for applications like video streaming and may lead to inaccurate performance estimations.

Reinforcement learning (RL) - based solutions have also been explored in UAV scenarios, such as in [4], [6], [7]. Although these studies provide valuable insights, they face challenges when addressing more realistic environments characterized by multiple continuous and discrete interdependent decision variables and a high-dimensional search space. As a result, they often abstract several influencing factors, address only a limited subset of the overall problem or translate the variables into a single domain (either discrete or continuous). For instance, [4] adopts a deep RL approach with an action space composed solely of a binary array defining the node type (edge or cloud) for task processing and the CPU fraction to be allocated, whereas [6] employs the Deep Deterministic Policy Gradient (DDPG) algorithm for UAV trajectory optimization, with an action space defined by the offloading decision, UAV velocity, and horizontal deflection angle. In turn, [7] proposes an Actor–Critic method to determine the UAV's flight angle and speed, task offloading ratio, and the user to be served. Moreover, the high dynamicity of 3D networks may require frequent retraining of RL models to adapt to varying scenarios, as well as careful hyperparameter tuning to prevent performance degradation, which may introduce additional time and computational overheads.

This paper addresses these gaps by proposing an evolutionary solution for task offloading in 3D Edge Computing Networks. Unlike RL approaches that require extensive training, parameter tuning, and retraining under dynamic conditions, our model-free, population-based optimization scheme can

This study was financed by the UFPE/ Propesqi via Edital Nº 06/2024. Caio Souza is also with Universidade de Pernambuco (UPE)

jointly handle both discrete and continuous decision variables. It simultaneous optimizes subtask partitioning and replication, node selection, and resource assignment, while accounting for failures in computation and bidirectional transmissions, as well as result fusion at the user device. It is evaluated under different task dimensions, partition counts, and replication levels. Results show that it achieves consistent performance across varying task complexities, effectively balancing latency, reliability, energy consumption, and efficiency. It attains high reliability (up to 0.9998 with increased replication) and reduces energy consumption compared to existing alternatives. Furthermore, the support for task partitioning and subtask replication enables efficient parallelism, enhancing both efficiency and reliability. Overall, our solution outperforms baseline methods when jointly considering the tradeoffs among latency, energy consumption, system robustness, and efficiency. The remainder of this paper is organized as follows. Section II presents the system model and formulates the task offloading problem. Section III details the proposed solution. Section IV discusses the results. Finally, Section V concludes the paper and outlines some future research directions.

II. System Model

The 3D edge computing network 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 device (UE) resources, the 3D system comprises M terrestrial MEC nodes, V UAVs, and E XEC nodes. From the user perspective, $\mathbb{K} = \{k_j\}$, with $j = 0, 1, ..., M, M+1, M+2, ..., M+V, M+V+1, ..., M+V+E$, denotes the set of available nodes that can be used to process tasks. 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 XEC nodes. Each node k_j is characterized by a CPU frequency f_j, a capacitance coefficient z_j, and a reliability level r_j. The system comprises U users, u_i. At time t, each user u_i has a task $T_i(t)$, with $T_i \in \mathbb{T}_t$, that involves processing $D_i(t)$ bits, requiring $Q_i(t)$ CPU cycles per bit. Since the UE and XEC nodes may concurrently run other applications from their owners, the computational resources available to process a task $T_i(t)$ can fluctuate over time. This availability is represented by $\theta_j(t)$, where $j = 0, M+V+1, \ldots, M+V+E$ denotes the node k_j. For the remaining remote nodes, we assume full resource availability for offloaded tasks, i.e., $\theta_j(t) = 1$.

We consider that a user task $T_i(t)$ can be partitioned into subtasks, replicated, and processed across multiples nodes, including the UE. Moreover, UAVs are modeled as stationary nodes within each time slot. In addition, variations in communication conditions induced by mobility between consecutive time slots are captured through fluctuations in the SNR. This strategy can improve latency and reliability, but at the expense of higher energy consumption and resource usage. After the remote processing of subtasks, the results are returned to the UEs for fusion. We account for bidirectional transmissions and possible failures during computation and transmission, which may affect task reliability. To mitigate intracellular interference caused by simultaneous transmissions to different remote nodes, we consider that these nodes operate on orthogonal frequencies [8]. A centralized approach can be adopted to manage task offloading, where a controller receives user offloading requests, which include task characteristics (e.g., size and complexity), the computational capabilities of both UEs and remote nodes (e.g., available resources), and the wireless channel conditions. Based on this information, it determines the task-to-node mapping and allocates the corresponding computational and communication resources. Once the allocation plan is defined, UEs offload their tasks to the designated nodes, which subsequently process them using the assigned resources.

A. Computing Model

Given a task $T_i(t)$ divided into n_i disjoint subtasks $s_{ic}(t)$ of $d_{ic}(t) = \phi_{ic}(t)D_i(t)$ bits each, where $c = 1, 2, ...n_i$, $n_i \in \mathbb{N}$, $\phi_{ic}(t) \in \mathbb{R}$, $0 < \phi_{ic}(t) \leq 1$, $\sum_{c=1}^{n} \phi_{ic}(t) = 1$, and that some subtasks are replicated to enhance reliability, where $\mathbb{S}_{ic}$ denotes the set of replicas of subtask $s_{ic}(t)$, $q_{ic} = |\mathbb{S}_{ic}|$, with $\sum_c q_{ic} \leq |\mathbb{K}|$, and $\mathbb{K}_i(t) \subseteq \mathbb{K}$ being the set of nodes (UE/remote) selected to process the user u_i subtasks/replicas at time t, the latency to process the subtask s_{ic} or its replica assigned to the node $k_j \in \mathbb{K}_i(t)$ is given by Eq. 1, where $o_{icj}(t)$ is a binary variable that equals 1 if node k_j is assigned to process subtask s_{ic} of user u_i at time t. $\alpha_{ij}(t) \in \mathbb{R}^+$, with $|\alpha_{ij}(t)| \leq 1$ and $\sum_i \alpha_{ij}(t)| \leq 1$, denotes the CPU fraction of the node k_j allocated to this processing.

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

When subtasks are processed on remote nodes, their results need to be merged to build the final output. We consider that this fusion occurs at the UE side (k_0), which incurs an additional delay. Let $\delta_i(t)$, $0 < \delta_i(t) \leq 1$, represent the ratio between the amount of data offloaded to remote nodes and the amount returned to the UE for consolidation. The latency for data fusion of task T_i at time t is given by $L_{f_i}(t)$, where the summation includes only remote nodes. Similar to $D_i(t)$ and $Q_i(t)$, the value of $\delta_i(t)$ reflects the application's nature and can be determined by profiling the applications.

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

The energy spent, in Joules (J), to process a subtask s_{ic} on node k_j is given by Eq. 3. The total energy consumption for task T_i is expressed by Eq. 4, which accounts for all subtask replicas, applies Eq. 3 to compute $Ep_{ic'}(t)$, and incorporates the energy consumed in the data fusion process.

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

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

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

Hardware or software failures may occur during task processing. To account for these, we consider the node reliability r_j, defined as the complement of its failure probability. It can be estimated using the mean time between failures and the mean time to repair, which are typically derived from historical failure and maintenance records [9]. Since task $T_i(t)$ may be processed across multiple nodes with admitted redundancy, and letting $\mathbb{S}_{ic}$ denote the set of copies of the subtask s_{ic}, the overall processing reliability of T_i is given by Eq. 15.

B. Communication Model

The bidirectional transmission assumes adaptive modulation and coding scheme (MCS) based on the channel quality. It relies on an indicator (CQI) periodically reported by the UE to the base station, which dynamically adjusts the MCS to maintain the block error rate (BLER) below a threshold. This work follows the 3GPP TS 38.214 specification (version 16.2.0, Release 16) to represent the channel quality through a CQI value and map it to the corresponding spectral efficiency (SE, in bit/s/Hz), indexing the MCS table, as in [10].

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} refers to the subtask size in bits and and δ_i denotes the ratio of data returned to the UE. $V_{up_{ij}(t)}$ and $V_{down_{ij}(t)}$ represent the uplink and downlink data rates between the user u_i and the node k_j, as obtained by Eq. 7, where the transmission block size (TBS) is computed following 3GPP TS 38.214 version 16.2.0 Release 16, as in [10], and OH is the physical downlink/uplink shared channel (PDSCH/PUSCH) overhead.

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

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

The number of resource blocks (RBs) for downlink or uplink communications is given by Eq. 8, where $\beta_{ij}(t) \in [0, 1]$ represents the fractional bandwidth allocated to uplink or downlink transmissions between user u_i and node k_j, and B denotes the total system bandwidth.

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

Given a subtask $s_{ic}(t)$ replicated into $q_{ic}(t)$ copies, its latency $L_{ic}(t)$, which encompasses uplink, downlink, and processing times, is defined as the minimum individual latency $L_{ic'}(t)$ among all replicas $s_{ic'}(t) \in \mathbb{S}ic$, as shown in Eq. 9. $Lic'(t)$ is given by Eq. 10, where for copies $s_{ic'}(t)$ processed on the UE, $L_{up_{ic'}}(t)$ and $L_{down_{ic'}}(t)$ are zero. The total latency for completing task T_i, including the data fusion delay, is given by Eq. 11.

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

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

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

The transmission reliability refers to the successful transfer of data packets over a communication link. The transmission failure probability Pf_{ic} of the subtask c is given by Eq. 14, where $Pf_{ic'_{up}}(t)$ and $Pf_{ic'_{dw}}(t)$ represent the uplink and downlink failure probabilities of the copy c', and TBS_{up} and TBS_{dw} are the uplink and downlink TBSs, respectively, as in [10]. 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 a reliability purpose, subtasks (replicas) of the same task are not allowed to be processed on the same node.

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

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

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

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

$$R_i(t) = Rp_i(t) Rt_i(t) \tag{16}$$

The energy consumption for uplink and downlink of T_i considers the user (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 shown in Eqs. 18 and 17. The total energy consumed to offload T_i is given by Eq. 19, which accounts for energy spent on the task uplink, downlink, processing, and fusion.

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

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

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

C. Problem Formulation

Deciding where processing a task and the amount of resource to be used impact its latency, reliability, and energy consumption. In this context, the following task offloading problem can be formulated. 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 allocation to minimize the average energy consumption ($\overline{E}(t)$) and latency ($\overline{L}(t)$) while maximizing the average reliability ($\overline{R}(t)$). Formally,

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

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

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

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

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

III. A Genetic Algorithm (GA)-based Solution

A Genetic Algorithm (GA) was designed to solve the problem in Eq. 20. It evolves a set of candidate solutions, represented by chromosomes, across generations, via operators such as selection, crossover, and mutation, to find high-quality solutions [11]. Its versatility, parallelism, and computational simplicity make it well-suited for addressing the problem in Eq. 20, as the GA operates directly on the native domains of decision variables without requiring translation into a single domain, which could otherwise degrade solution quality.

A. Chromosome Structure

In our solution, each chromosome m, represented by $X^m = [x_1^m, x_2^m, x_3^m, ...x_l^m, ...x_b^i,]$, is composed of b genes. For $1 \leq l \leq n_i$, x_l^m represents the data fraction (ϕ_{ic}) associated with each substask of the task T_i, with $x_l^m \in$ (0 1) and $\sum_l x_l^m = 1$. For $n+1 \leq l \leq n_i(1+q_i)$, x_l^m is a binary value that indicates the existence of a replica for a given subtask of T_i, where each subtask may have up to q_i copies, represented by a binary sequence of q_i bits. The next part of the chromosome encodes the node allocation for the subtasks, including their replicas. Specifically, for $n_i(1+q_i)+1 \leq l \leq 2n_i(1+q_i)$, $x_l^m \in \mathbb{K}$ represents a processing node (k_j) assigned to handle a given subtask of T_i. The computing and communication resources allocated for processing and offloading the subtasks in their respective assigned nodes are represented in the subsequent genes of the individual. Thus, $x_l^m \in$ (0 1), with $2n_i(1+q_i)+1 \leq l \leq 3n_i(1+q_i)$, denotes the CPU portion of the node assigned to process a given subtask. For $3n_i(1+q_i)+1 \leq l \leq 4n_i(1+q_i)$ and $4n_i(1+q_i)+1 \leq l \leq b = 5n_i(1+q_i)$, x_l^m quantifies the bandwidth, expressed as the number RBs (see Eq. 8), allocated for uplink and downlink , respectively.

B. Fitness Function

The fitness is evaluated based on latency, reliability, and energy consumption derived from the chromosome. To reduce the multiobjective problem (Eq. 20) complexity, the fitness function (Eq. 25) incorporates the average latency, energy consumption, and reliability, each normalized by its respective maximum average value (Eq. 26), into an exponential formulation. This approach increases the sensitivity to small variations, which is further scaled by a factor ζ. i and k represent indexes for task (a total of N) and chromosome, respectively.

$$F(X^m) = \zeta e^{(w_l \frac{1}{1+\hat{L}_m} + w_e \frac{1}{1+\hat{E}_m} + w_r \frac{1}{(2-\hat{R}_m)})} \tag{25}$$

$$\hat{L}_m = \frac{\sum_i L_i(t)/N}{\max_k\{\hat{L}_k\}}; \hat{E}_m = \frac{\sum_i E_i(t)/N}{\max_k\{\hat{E}_k\}}; \hat{R}_m = \frac{\sum_i R_i(t)/N}{\max_k\{\hat{R}_k\}} \tag{26}$$

C. Genetic Operators

The roulette wheel method, based on fitness values, was used to select individuals for the crossover process [11]. Since the chromosome encodes the overall solution in distinct segments corresponding to subproblems (e.g., task partitioning, replica decision, node allocation, and resource allocation)and each segment must adhere to specific constraints on the values it may contain, different crossover operators for the segments were adopted, with adaptations to ensure the resulting offspring satisfy the problem constraints. For the task partitioning segment, the blended crossover (BLX-α) [11] is applied with $\alpha = 0.5$ and $\beta \sim (-\alpha, 1+\alpha)$, where the offspring's genes ($x_l^{m_1}$ and $x_l^{m_2}$, with $1 \leq l \leq n_i$) are determined by combining the parents ($x_l^{p_1}$ and $x_l^{p_2}$) via Eq. 27

For the second and third segments (replica indication and node assignments), the uniform crossover is applied [11]. For node assignments, must ensure that subtasks of the same task are not allocated to the same node, i.e., $x_l^m \neq x_k^m, \forall l \neq k$, where $n_i(1+q_i)+1 \leq l,k \leq 2n_i(1+q_i)$. For CPU allocation genes, two crossover operators are employed: BLX-α and uniform. The former (Eq. 27) determines the CPU fraction for original subtasks, while the latter handles replicas, some of which may be absent, to avoid unnecessary node or resource allocation. In both cases, the assigned CPU fraction cannot exceed the node's available capacity. For uplink and downlink bandwidth allocation segments, the uniform crossover is used, with a constraint ensuring that the bandwidth assigned for task offloading remains within available resources.

To improve search diversity and avoid local minima, bit and gene mutations [11] are applied to the second and third segments, respectively. The former flips a binary gene, whereas the latter replaces the current node with an unused one. For other segments, Michalewicz's mutation perturbs the gene value according to Eq. 28, using $P = 5$ and Y_l set to 0.4 and 5 for CPU and bandwidth, respectively.

$$x_l^{m_1} = |x_l^{p_1} + \beta(x_l^{p_1} - x_l^{p_2})|; x_l^{m_2} = |x_l^{p_2} + \beta(x_l^{p_2} - x_l^{p_1})| \tag{27}$$

$$x_l^m = x_l^m + Y_l(1 - \xi^{(1-\epsilon/G)^P}) \tag{28}$$

D. GA Flow Execution

Initially, a population of solutions is randomly generated. Each one is then evaluated based on the fitness function (Eq. 25). Next, selection, crossover, and mutation operators are applied, with an elitist strategy ensuring that the best individuals are preserved across the generations. After forming the new population, the stop criterion (number of generations (G)) is evaluated. If it is not met, the process repeats; otherwise, the best individual is selected as the final solution.

IV. Result Analysis

We evaluate our solution under three scenarios. The first scenario (IV-B) varies task dimensions, using three sets of task size (D_i) and complexity (Q_i) values to represent low (L), moderate (M), and high (H) dimensionality tasks [1]. The second scenario (IV-C) examines different task partition levels to assess the impact of parallelism. The third scenario analyzes performance when varying the maximum number of subtask replicas. Table I summarizes the parameter values used across all scenarios. Results are presented as averages over 50 runs with a 95% confidence level. The GA parameters—population size, number of generations, mutation probability, and crossover probability—were set to 100, 100, 0.01, and 0.7, respectively. We assess our solution in terms of latency, reliability, energy consumption, and efficiency, where efficiency measures the number of bits handled (transmitted/processed) per joule (J). The performance is compared

against two schemes: Random, where all decision variables are chosen randomly and tasks are mapped in parallel; and Ateya, which follows [12] by mapping tasks sequentially, assigning each task to a single node without considering partitioning or replication. Node selection prioritizes the option that minimizes both latency and energy consumption, falling back to latency if no suitable node is found.

A. GA convergence

Before evaluating the GA under the three scenarios, its convergence behavior was analyzed in terms of the maximum fitness value across generations. Scenario 2 with with $n = 4$ was considered, and the evolution of the maximum fitness is shown in Fig. 1. As observed, the fitness value increases during the first 40 generations, reflecting the GA's exploration phase, in which new solutions are generated, and the search space is extensively explored. In the next generations, the best solution is achieved, and the maximum fitness stabilizes, indicating convergence. Although other GA-based resource allocation approaches in the literature exhibit longer convergence times, it is worth noting that our solution converges relatively quickly, demonstrating its feasibility for task offloading problems.

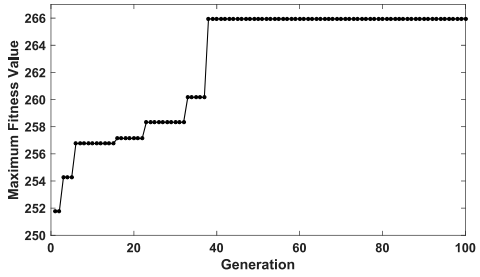


Fig. 1: The Maximum Fitness Evolution across Generations.

B. Task Dimension Scenario (D_i, Q_i)

Fig. 2a shows a promising performance of the GA solution when handling tasks of varying dimensionalities, achieving latencies of 0.660 s (L), 6.889 s (M) and 28.342 s (H) while the Random one results in latencies of 0.654 s, 7.881 s, and 39.515 s, respectively. Although the Random approach performs similarly for low-dimensional tasks, it becomes significantly less efficient as complexity increases. On the other hand, the Ateya scheme, which prioritizes low latency and does not adopt task partitioning, not spending time on data fusion, achieves the best performance, with values of 0.328 s (L), 3.746 s (M) and 19.547 s (H). However, this comes at the cost of higher energy consumption, as well as reduced reliability and efficiency, as shown in Figs. 2b, 2c, and 2d, where the Ateya boosts the energy consumption by up to 583% and causes a reduction in reliability and efficiency of up to 0.1 and 95% when compared to our GA solution in the moderate case. The GA scheme, in turn, exhibits an almost consistent behavior across increasing task dimensionalities, with high reliability values of 0.9994 (L), 0.9983 (M), and 0.9950 (H). This behavior aligns with expectations, as the GA scheme incorporates subtask replication, with replicas being processed on distinct nodes, ensuring redundancy. Although it performs worse than GA, the Random scheme leverages parallelism and redundancy to outperform Ateya in terms of reliability, energy consumption, and efficiency, achieving a gain of up to 0.05, 62%,and 88%, respectively, for high-dimensionality tasks.

TABLE I: Simulation Parameters

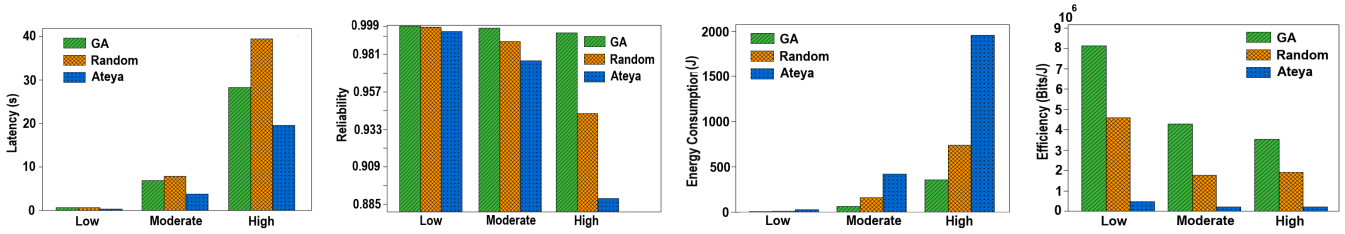
Scenario		**D (MB)**	**Q(cycle/bit**	**n**	**q**
Dimension	L	(0.6,2)	(60, 100)	4	4
	M	(5,10)	(120, 240)	4	4
	H	(20,60)	(120, 240)	4	4
Parallelism		(0.6, 2)	(60, 100)	2,4,6	4
Replication		(0.6, 2)	(60, 100)	4	2,4,6
Parameters		User	Ext.	Uav	Mec
# Nodes		10	20	15	15
f_j (GHz)		(1,2)	(1,2)	(1,3)	(20,25)
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)	1	1	(0.7,1)
$SINR$ (dB)		(10,25)			
$nPRB$		100 (Uplink), 100 (Downlink)			
$BLER$, μ, δ_i		0.00001, 3, (0.001, 0.4)			

C. Task Parallelism Scenario (n)

Increasing the number of partitions leads to a slight reduction in latency in the GA approach, indicating that higher parallelism in data transmission and reception reduces the average processing time per task, as shown in Fig. 3a. However, transmitting more partitions simultaneously across nodes also introduces potential failure points, resulting in a marginal drop in system reliability (about 0.05% lower when comparing $n = 2$ and $n = 6$). Fig. 3b confirms that the GA solution consistently maintains reliability above 0.999 across all scenarios. Although the Ateya scheme lower latency, Figs. 3c and 3d reveal that it significantly compromises energy efficiency. In contrast, the proposed solution shows improved efficiency for higher parallelism level. Notably, increasing the number of partitions from 2 to 4 enhances energy efficiency by 19%, while the jump from 2 to 6 yields only an 8.6% gain. This diminishing return stems from the energy cost associated with the increased number of simultaneous transmissions and receptions, which offsets the benefits of selecting energy efficient nodes. In Figs. 3b, 3c, and 3d, we observe slight variations in Ateya's performance as n changes. Since Ateya does not employ task partitioning, these variations result from the sequential manner in which it maps the tasks, where the order impacts performance. Similar to the previous scenario, the Radom scheme presents an intermediate performance regarding reliability, energy consumption, and efficiency.

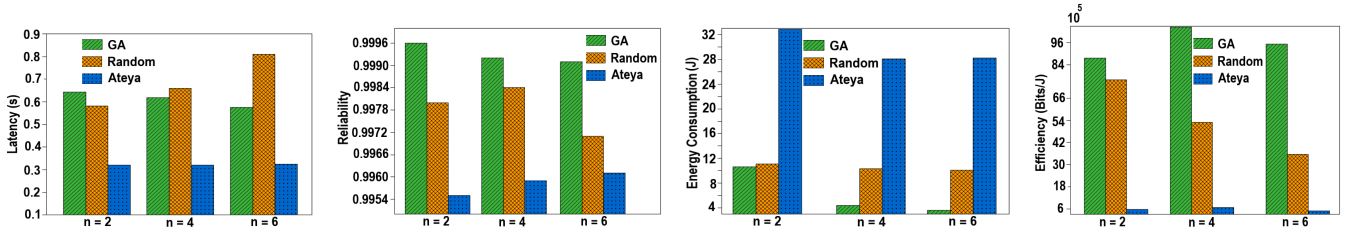
D. Task Replication Scenario (q_i)

Figs. 4b and 4d show that both the reliability and efficiency of the GA approach improve as the maximum number of replicas (q) increases, with reliability rising from 0.9978 ($q = 2$) to 0.9998 ($q = 6$). This behavior is expected, as increasing subtask replication enhances redundancy and fault tolerance. Moreover, as shown in Figs. 4c-4d, higher replica availability leads to a reduction in energy consumption, dropping by approximately 1.2 J from $q = 2$ to $q = 6$. This



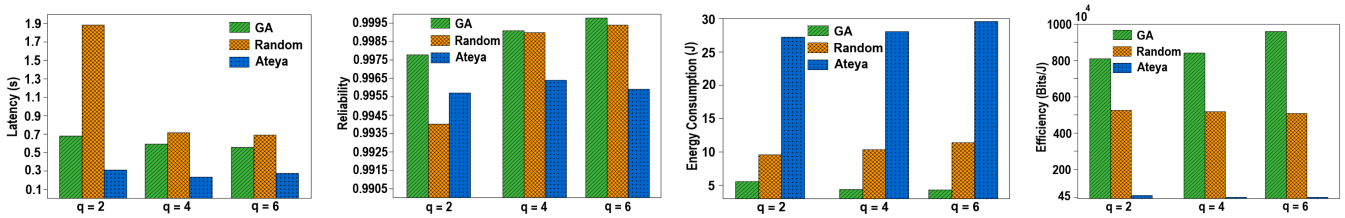
(a) Latency. (b) Reliability. (c) Energy Consumption. (d) Efficiency.

Fig. 2: Results varying the task dimensionality.



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

Fig. 3: Results varying the number of partitions for tasks (parallelism level.)



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

Fig. 4: Results varying the maximum number of replicas for subtasks.

occurs because the system, with greater replication flexibility, can allocate replicas to less reliable but more energy-efficient nodes while still maintaining overall reliability. Similarly, Fig. 4a shows that this elasticity enables the system to prioritize nodes with lower latency, relying on replication to compensate for potential reliability losses. Ateya exhibits a similar pattern across all q values, as it does not employ replication, performing worse than GA and Random in most metrics except latency, its main focus.

V. Conclusion

This work addressed task offloading and resource allocation in 3D Edge Computing Networks by jointly optimizing task partitioning and replication, node selection, and resource allocation, while considering failures in computation and transmissions, and data fusion at the UE. The proposed solution outperformed the baselines, delivering consistent results across varying task complexities, parallelism levels, and replica counts, while balancing latency, reliability, energy consumption, and efficiency. As future directions, we suggest extending the framework to include satellite communications and exploring hybrid solutions that integrate the strengths of machine learning and evolutionary computing.

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] L. Lin, X. Liao, H. Jin, and P. Li, "Computation offloading toward edge computing," *Proc. of the IEEE*, vol. 107, no. 8, pp. 1584–1607, 2019.

[3] E. E. Haber, H. A. Alameddine, C. Assi, and S. Sharafeddine, "A reliability-aware computation offloading solution via uav-mounted cloudlets," in *2019 IEEE 8th International Conference on Cloud Networking (CloudNet)*, 2019, pp. 1–6.

[4] 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.

[5] Y. Li, D. V. Huynh, T. Do-Duy, E. Garcia-Palacios, and T. Q. Duong, "Unmanned aerial vehicle-aided edge networks with ultra-reliable low-latency communications: A digital twin approach," *IET Signal Processing*, vol. 16, no. 8, pp. 897–908, 2022.

[6] J. Zhang, G. Zhang, X. Wang, X. Zhao, P. Yuan, and H. Jin, "Uav-assisted task offloading in edge computing," *IEEE Internet of Things Journal*, vol. 12, no. 5, pp. 5559–5574, 2025.

[7] 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*, vol. 13, pp. 68 710–68 725, 2025.

[8] 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.

[9] 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.

[10] 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 *XIV Brazilian Symposium on Computing Systems Engineering (SBESC)*, 2024, pp. 1–6.

[11] Z. Michalewicz, *Genetic algorithms + data structures = evolution programs (3rd ed.)*. Berlin, Heidelberg: Springer-Verlag, 1996.

[12] 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.