DRL-based Trajectory Planning and Sensor Task Scheduling for Edge Robotics

Sirine Bouhoula*, Marios Avgeris*, Aris Leivadeas*, Ioannis Lambadaris†
* *Department of Software and IT Engineering, École de technologie supérieure (ÉTS), Montréal, Canada*
†*Department of Systems and Computer Engineering, Carleton University, Ottawa, Canada*
sirine.bouhoula.1@ens.etsmtl.ca, marios.avgeris@etsmtl.ca, aris.leivadeas@etsmtl.ca, ioannis@sce.carleton.ca

***Abstract*—Mobile Edge Computing (MEC) and Edge Robotics have recently emerged as transformative technologies, revolutionizing industries by enabling real-time processing, decision-making, and automation at the network edge. However, the dynamicity induced by the system's conditions and specifically the mobility poses a challenge for optimally deciding where to execute a given computational task. As a response, we develop an intelligent algorithm for dynamic sensor task offloading tailored to the unique requirements of MEC-enabled robotic environments. Specifically, we first introduce the environmental dynamics including a sensor task's end-to-end delay and the robots' mobility and energy consumption and provide mathematical formulations to model these dynamics. Then, we mathematically formulate the optimization problem and its MDP counterpart and we propose a Deep Reinforcement Learning (DRL)-based computational offloading strategy to jointly optimize Quality of Service (QoS) and energy consumption through robot trajectory planning and fine-grained task allocation. Through hand-picked representative simulation scenarios, we demonstrate the superiority of our proposed mechanism in enhancing the overall system performance, specifically in optimizing task execution, reducing energy consumption, and mitigating transmission delays, compared to various baseline approaches.**

***Index Terms*—Mobile Edge Computing, Edge Robotics, Task Offloading, Trajectory Planning, Deep Reinforcement Learning.**

I. Introduction

In the dawn of 5G and beyond networking, Mobile Edge Computing (MEC) has emerged as a transformative paradigm, empowering real-time processing and decision-making at the network edge. This shift from traditional Cloud-centric approaches offers significant advantages, including ultra-low latency, high bandwidth, and improved reliability [1]. Building upon this foundation, in the realm of the Internet of Things (IoT), edge robotics integrates robotic systems and sensors with the MEC infrastructure. Following the paradigm of *computational task offloading*, this powerful combination enables robots to execute complex tasks by leveraging additional computational and networking resources in their proximity. Edge robotic systems hold great potential across various domains, including industrial automation, logistics, and search-and-rescue operations [2].

This work has been supported by the Natural Sciences and Engineering Research Council of Canada (NSERC), Grant no. RGPIN-2019-05250

However, environments like these pose significant challenges for MEC-enabled edge robotic systems due to their inherent dynamicity. Mobility, for instance, can impact the overall system performance. As robots navigate within an environment, the distance between them, the sensors they are associated with, and the targeted edge execution platforms can fluctuate, directly affecting data transmission delays and the overall Quality of Service (QoS). Furthermore, additional delays during task execution may arise due to factors such as network bandwidth and resource over-utilization. Thus, minimizing these delays is crucial for time-sensitive edge robotic assisted applications. Additionally, since robots operate on constrained resources and battery capacity, careful decision-making regarding local processing versus remote offloading is necessary to balance energy efficiency with QoS [3].

The problem of computational offloading in general falls into the category of NP-hard problems, thus approximate solutions have been suggested in the literature; specifically, Deep Reinforcement Learning (DRL)-based techniques have shown promising potential to delivering near-optimal results in highly dynamic environments, due to their model-free, scalable and generalizable operation [4]. As a response, to address the aforementioned challenges and harness the potential of edge robotic systems, we thus propose a DRL-based sensor computational offloading strategy to jointly optimize QoS and energy consumption through robot trajectory planning and fine-grained task allocation. Our contribution is threefold:

- We develop of a comprehensive mathematical model that effectively captures the dynamic aspects of the system like mobility patterns, transmission and execution delays, and energy consumption dynamics. We then formally formulate the corresponding optimization problem.
- Since this problem is challenging to solve in real time, we formulate its Markov Decision Process counterpart and propose a novel DRL-based online solution. Specifically, we design a mechanism where robots are able to jointly and dynamically i) determine whether to perform tasks locally or offload them to other robots in the vicinity, in a Device-to-Device (D2D) fashion, and ii) plan their trajectory based on their task execution strategy and the distance from associated sensors.
- Through extensive evaluation we benchmark the per-

formance of our solution against several baseline algorithms, demonstrating its superiority in terms of task execution, energy consumption, and overall performance.

The remainder of this paper is structured as follows: Section II provides a review of the related works. Section III details the system model, the problem formulation and the MDP components. Section IV proposes our DRL-based task offloading strategy for edge robotics, while Section V presents the simulation results. Finally, Section VI concludes the paper.

II. Related Works

Various approaches have been proposed to optimize computation offloading strategies in the context of MEC [5]. Many of these solve a snapshot of the problem by considering static mobile devices static during the optimization horizon. For example, Jiang *et al.* [6] presented a delay-aware energy minimization D2D offloading scheme based on dynamic programming, emphasizing the balance between energy consumption and delay constraints. While effective in addressing energy concerns, this approach lacks adaptability in mobile scenarios due to its limited consideration of dynamic mobility patterns. Wang *et al.* [7] focused on energy and delay minimization in D2D-assisted MEC systems. Although effective in reducing energy consumption and delay for partial offloading scenarios, the approach may require adaptations for varying network conditions and mobility. Li *et al.* [8] investigated energy-efficient D2D-assisted computation offloading in MEC systems with energy harvesting capabilities. While shown to prolong mission duration, this design overlooks dynamic mobility patterns. Lastly, Lalwani *et al.* [9] optimized resource allocation for crowd-cloud assisted D2D computation offloading, offering reduced latency. However, once again mobility is not taken into consideration in this work.

On the other hand, Jeon *et al.* [10] considered mobility-aware task offloading in a distributed edge environments. The authors proposed an impractical optimization algorithm with high computational complexity while energy consumption was not considered. Li *et al.* [11] focused on minimizing costs in a mobility-aware task placement in D2D-assisted MEC environments. However, overlooking energy and delay optimization limits its applicability in resource-constrained environments. Fragkos *et al.* [12] introduced an AI-empowered scheme for UAVs data offloading to MEC servers. However, mission prolonging through energy minimization was not considered. Finally, a mobility-aware computation offloading scheme for swarm robotics, was presented in [13] leveraging DRL techniques. However, the mobility is only considered passively and is not included in the DRL action.

In contrast to existing works, our proposed approach utilizes a dynamically adapting DRL algorithm with mobility awareness that jointly decides where to offload and where to move to comprehensively optimize energy and delay in MEC environments. By considering both mobility patterns and computational efficiency, we aim to address the limitations of prior approaches and provide an effective computation offloading solution in dynamic and resource-constrained scenarios.

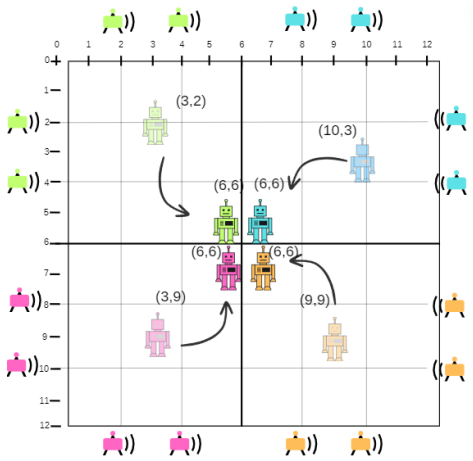


Fig. 1: System Overview.

III. System Modeling

We assume that the system operates within a controlled environment, providing a dynamic space for robots to navigate and perform tasks. Specifically, $\mathbb{I}$ wall-mounted sensors are strategically distributed throughout the environment, continuously collecting data and transmitting them for processing to one of the $\mathbb{R}$ mobile robots they are assigned to. Each robot $r \in \mathbb{R}$ is wirelessly tethered to a subset of $\mathbb{I}_r$ sensors. Receiving sensor data generates a number of computational tasks for the robots, which then need to decide where to execute the computation of said tasks, locally or on another available robot, and whether to relocate or not to facilitate energy- and delay-efficient task completion. Such a setup resembles a factory floor automation for *Industry 4.0/5.0* environments (e.g., Warehouse Robotics) where computationally hungry tasks include object detection for navigation and quality control among others [14]. An illustrative overview of the envisioned system setup is depicted in Fig. 1. The rest of this section outlines the system modeling and the formulation of the optimization problem. Then, the problem at hand is formulated as a Markov Decision Process (MDP) and the definitions for the main components are given.

A. Robot's Resources & Decisions

We assume that time is slotted in timeslots $t = 0, 1, \dots \mathbb{N}$. Each robot $r \in \mathbb{R}$ is equipped with C_r^{max} computational resources and E_r^{max} remaining energy. In our model, the robots operate on a discrete grid, where each cell represents a unit distance. The coordinates of each cell can be represented as integer pairs (x, y) with $x, y \in \mathbb{N}$. Mobility decisions are aimed at positioning each robot strategically within the environment to optimize task execution. Moving closer to sensors reduces the delay in receiving information and processing tasks, improving the overall system efficiency. On the other hand, if the offloading decision involves transferring tasks

to another robot, the moving robot may adjust its position to minimize transmission delay to other robots, ensuring efficient task execution. However, the wireless robot-sensor tethering has a distance limit W_{max}, so their mobility is constrained by it. Let each robot initiate from a position (x_r^0, y_r^0) on the grid and at each timeslot t it is allowed to move to neighboring cells that are adjacent to its current position in the cardinal directions: up, down, right, or left, or remain in the same position. Since each robot's movement is assumed to be controlled in our environment, we introduce the following decision variable $m_r^{(t)} = (m_{r_x}^{(t)}, m_{r_y}^{(t)})$, with $m_{r_x}^{(t)}, m_{r_y}^{(t)} \in \{-1, 0, 1\}$ subject to $m_{r_x}^{(t)} + m_{r_y}^{(t)} \leq 1$. Then, each robot's movement can be described by the following discrete-time process:

$$x_r^{(t+1)} = x_r^{(t)} + m_{r_x}^{(t)}, \tag{1}$$

$$y_r^{(t+1)} = y_r^{(t)} + m_{r_y}^{(t)}. \tag{2}$$

To accommodate the offloading decision process, we introduce the binary decision variable $o_{r,r'}^{i(t)} \in [0, 1]$, which equals to 1 if robot r offloads the task i to another robot r'. Without loss of generality, we assume that the task can be only offloaded once, by the initial robot-receiver. Specifically, each robot receives one task at the beginning of a timeslot, while the movement, offloading decision, as well as the execution of the task are completed within said timeslot. For ease of presentation, in the following we temporarily drop the timeslot notation t.

B. End-to-end Delay

1) Transmission Delay: We assume that at each given point, each sensor $i \in \mathbb{I}$ transmits data regarding a single task. The transmission delay, $D_{i,r}^T$, incurred during information exchange is affected by the data size, L (in KB), available bandwidth, b (in Hz), channel gain, h, and noise power of the communication channel, σ^2 (in dBm), as well as the transmit power P (in dBm). When a robot $r \in \mathbb{R}$ receives data from a sensor, the transmission delay depends on the distance between the robot and sensors, denoted by d_r^i. If subsequently the task is offloaded to another robot $r' \in \mathbb{R}$, there is an additional transmission delay incurred, which depends on the distance between the robots, denoted by $d_{r,r'}$. Hence, the total transmission delay for task i generated at timeslot t for robot r is given according to the Shannon-Hartley theorem by:

$$D_{i,r}^T = L_i \left(\frac{d_{i,r}}{b \log_2\left(1 + \frac{P_i\, h}{|\sigma^2|}\right)} + \sum_{r'=1}^{|\mathbb{R}|} o_{r,r'}^i \frac{d_{r,r'}}{b \log_2\left(1 + \frac{P_r\, h}{|\sigma^2|}\right)} \right), \tag{3}$$

2) Execution Delay: The execution delay ($D_{i,r}^E$) for a task i is inversely proportional to the computational resources c_i required by the task i and allocated by robot r for its completion and is calculated as:

$$D_{i,r}^E = \frac{L_i}{c_i}. \tag{4}$$

3) End-to-end Delay: Subsequently, we define the task end-to-end delay ($D_{i,r}$) as the sum of the transmission and execution delays:

$$D_{i,r} = D_{i,r}^T + (1 - \sum_{r'=1}^{|\mathbb{R}|} o_{r,r'}^i) D_{i,r}^E + \sum_{r'=1}^{|\mathbb{R}|} o_{r,r'}^i D_{i,r'}^E. \tag{5}$$

C. Energy Consumption

1) Transmission Energy Consumption: We assume that the energy a robot r consumes for receiving data from a sensor i is negligible. Thus, the only transmission energy consumption induced on the robot's side is a result of offloading the task to robot r' and it depends on the size of the task L_i, the robot's transmission power P_r and the duration of the transmission, [15] as follows:

$$E_{i,r,r'}^T = P_r \frac{L_i\, d_{r,r'}}{b \log_2\left(1 + \frac{P_r\, h}{|\sigma^2|}\right)}. \tag{6}$$

2) Execution Energy Consumption: The execution energy consumption for a task i is proportional to the computational resources c_i required for its completion and is calculated as:

$$E_{i,r}^E = K\, c_i^2\, L_i, \tag{7}$$

where K is the effective switch capacitance coefficient that determines the relationship between the allocated computational resources and the energy consumed for executing task i.

3) Total Energy Consumption: The calculation of the system energy consumption at timeslot t for robot r is the sum of the transmission and execution energy consumption as a result of the offloading decisions, and is given by:

$$\begin{aligned} E_r = &\sum_{r'=1}^{|\mathbb{R}|} \sum_{i=1}^{|\mathbb{I}_{r'}|} o_{r',r}^{i,(t)} E_{i,r}^E + \\ &\sum_{r'=1}^{|\mathbb{R}|} \sum_{i=1}^{|\mathbb{I}_r|} (1 - o_{r,r'}^{i,(t)}) E_{i,r}^E + \\ &\sum_{r'=1}^{|\mathbb{R}|} \sum_{i=1}^{|\mathbb{I}_r|} o_{r,r'}^{i,(t)} E_{i,r,r'}^T \end{aligned} \tag{8}$$

D. Problem Definition

To prolong the mission duration and at the same time optimize task completion delays, we formulate the following optimization problem, which results in jointly finding the optimal trajectory and task scheduling decisions that minimize

the total system energy consumption and task completion delay over a time horizon T:

$$\min_{m_r^{(t)}, o_{r,r'}^{(t)}} \quad \sum_{t=1}^{T}(\sum_{r=1}^{|\mathbb{R}|}(\alpha E_r^{(t)} + \sum_{i=1}^{|\mathbb{I}_r|} D_{i,r}^{(t)})) \tag{9a}$$

$$\text{s.t.} \quad \sum_{r'=1}^{|\mathbb{R}|} o_{r,r'}^{i,(t)} \leq 1, \ \ \forall r \in \mathbb{R}, \forall i \in \mathbb{I}_r, \forall t \in T, \tag{9b}$$

$$m_{r_x}^{(t)} + m_{r_y}^{(t)} \leq 1, \ \ \forall r \in \mathbb{R}, \forall t \in T, \tag{9c}$$

$$D_{i,r}^{(t)} \leq D_i^{max}, \ \ \forall r \in \mathbb{R}, \forall i \in \mathbb{I}_r, \forall t \in T, \tag{9d}$$

$$\sum_{r'=1}^{|\mathbb{R}|}\sum_{i=1}^{|\mathbb{I}_{r'}|} o_{r',r}^{i,(t)} c_i^{(t)} + \sum_{r'=1}^{|\mathbb{R}|}\sum_{i=1}^{|\mathbb{I}_r|}(1 - o_{r,r'}^{i,(t)}) c_i^{(t)} \leq \leq C_r^{max}, \forall r \in \mathbb{R}, \forall t \in T, \tag{9e}$$

$$\sum_{t=1}^{T}\sum_{r=1}^{|\mathbb{R}|}\sum_{i=1}^{|\mathbb{I}_r|} E_r^{(t)} \leq E_r^{max}, \forall r \in \mathbb{R}, \forall t \in T, \tag{9f}$$

$$d_{i,r}^{(t)} \leq W_{max}, \forall r \in \mathbb{R}, \forall i \in \mathbb{I}_r, \forall t \in T. \tag{9g}$$

where $0 \leq \alpha \leq 1$ is a coefficient that introduces a bias for the energy consumption against delay minimization. Constraint (9b) ensures that each task is offloaded to at most one robot. Constraint (9c) restricts the movement of each robot to neighboring cells on the grid. Constraint (9d) imposes a maximum end-to-end delay limit for each task. Constraints (9e)-(9f) ensure that each robot's computational and energy resources are not depleted and finally, constraint (9g) limits the mobility of the robot to the proximity of its sensors. Since the calculation of energy and delay depends on the movements of the robots, and these movements are quantized in the grid, combinatorial aspects are introduced to the problem, as the movement decisions need to be made from a finite set of possible actions. Additionally, the non-linear dependencies posed as a result, further complicate the problem, making it difficult to devise an algorithm that provides real-time solutions. Instead, we decide to decompose the problem into a sub-problem to be solved at each timeslot $t \in T$ and formulate it as a Markov Decision Process. Then, we propose a DRL-based mechanism for solving it online.

E. Markov Decision Process (MDP) Formulation

In a typical MDP, the basic structure comprises five essential components: $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}\}$. Here, $\mathcal{S}$ refers to the *state space*, encompassing all the possible scenarios that a solver agent might encounter. The available set of *actions* for the agent is denoted by $\mathcal{A}$. Transition probabilities, represented by $\mathcal{P}$, quantify the likelihood of transitioning from one state to another given a specific action. Rewards obtained as a consequence of actions are denoted by $\mathcal{R}$. As the state space in our scenario is high-dimensional, precise calculation of $\mathcal{P}$ is not feasible. Thus, the MDP is simplified into a model-free process $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{R}\}$, defined as follows:

1) State: At each timeslot t, the system state $\mathcal{S}_t$ contains each robot's available computational and energy resources, $\dot{c}$ and $\dot{e}$ respectively, incoming tasks, ι assigned from its tethered sensors, and position $(x_r^{(t)}, y_r^{(t)})$ in the grid:

$$\mathcal{S}_t = \{\dot{c}_r^{(t)}, \dot{e}_r^{(t)}, \iota_r^{(t)}(x_r^{(t)}, y_r^{(t)}) \mid \forall r \in \mathbb{R}\}. \tag{10}$$

A state is terminal if there are no more tasks left to execute on any robot, i.e., $\iota_r^t == 0, \forall r \in \mathbb{R}$.

2) Action: We define a valid action as the joint decision of the robots' movements and their offloading choice regarding incoming tasks from sensors on the current timeslot t. First, each robot determines its movement, then, decides whether to execute tasks locally or offload them to other robots. This sequential process reflects the practical scenario where movement precedes task execution or offloading.

$$\mathcal{A}_t = \{o_{r,r'}^{i,(t)}, m_r^{(t)} \mid \forall r, r' \in \mathbb{R}, \forall i \in \iota_r^{(t)}\}. \tag{11}$$

3) Reward: After performing an action $\mathcal{A}_t$ on state $\mathcal{S}_t$, the agent gets a feedback, which directly determines its strategy. As our optimization objective is to minimize both the energy consumption and task completion delay, we formulate the reward as the scaled evaluation of the objective function Eq. (9). To embed the remaining tolerance constraints, the reward is severely penalized if the actions result in violation of the system constraints (9d)-(9g):

$$\mathcal{R}_t = \begin{cases} \ll 0, & \text{if } violation, \\ \Delta(\sum_{r=1}^{|\mathbb{R}|}(\alpha E_r^{(t)} + \sum_{i=1}^{\iota_r^{(t)}} D_{i,r}^{(t)}), & \text{otherwise,} \end{cases} \tag{12}$$

where Δ is a scaling constant coefficient.

IV. DRL-based Trajectory Planning and Sensor Task Scheduling for Edge Robotics

Having formulated an MDP for the problem (9), we utilize DRL to efficiently solve it. Specifically, we make use of a Deep Neural Network (DNN) to estimate $Q(\mathcal{S}_t, \mathcal{A}_t)$, i.e., the expectation of the long-term reward for each pair of system state and robots' decision. This value is typically calculated through the Bellman equation: $Q(\mathcal{S}_t, \mathcal{A}_t) \leftarrow Q(\mathcal{S}_t, \mathcal{A}_t) + \zeta(\mathcal{R}_t|_{\mathcal{S}_t, \mathcal{S}_{t+1}, \mathcal{A}_t} + \gamma \, max_{\mathcal{A}_{t+1}} Q(\mathcal{S}_{t+1}, \mathcal{A}_{t+1}) - Q(\mathcal{S}_t, \mathcal{A}_t))$, where ζ is the learning rate that satisfies $0 < \zeta < 1$, and $0 < \gamma < 1$ is the discount factor, used to model the uncertainty in the future actions. However, as the number of robots, sensors and the size of the grid becomes larger, the system's possible states and actions grow exponentially in size, making the exact computation of this Q-value in real time infeasible.

The basic idea behind training the DNN to estimate the Q-value by using a Deep Q-Networks (DQN) mechanism is briefly presented in Algorithm 1. Notice that we make use of an experience replay memory to improve the sample efficiency and stability of training, while we adopt the ϵ-greedy policy for the joint trajectory planning and task offloading action selection, since it gives us a better balance between exploration and exploitation. To update the weights of the

Q-network, we use the Stochastic Gradient Descent (SGD) algorithm. This approach enables real-time decision-making for the robots, without relying on a pre-defined system model.

After having the DNN's weights trained, the online phase of the proposed framework initiates with acquiring information about the robots' and the environment state. Based on this information, it calculates the current system state $\mathcal{S}_t$ using Eq. (10). With the current state available, the agent then selects an action $\mathcal{A}_t$ using the trained network's policy. This action reflects the joint robots' movement and task offloading decisions for the timeslot t.

Algorithm 1: DRL-based Trajectory Planning and Sensor Task Scheduling for Edge Robotics - Training

Initialize network Q and target network $\hat{Q}$ randomly.
Initialize experience replay memory $\mathcal{B}$.
for $t = 1, 2, \ldots$ **do** // Sampling
  $\epsilon \leftarrow$ set new epsilon with ϵ-decay.
  Compute current system state $\mathcal{S}_t$.
  Select a joint movement and offloading action for all robots $\mathcal{A}_t$ using ϵ-greedy policy.
  Execute $\mathcal{A}_t$, *produce* $\mathcal{S}_{t+1}$ and *collect* $\mathcal{R}_t$.
  $done \leftarrow \iota_r^{(t)} == 0, \forall r \in \mathbb{R}$
  Store experience $(\mathcal{S}_t, \mathcal{S}_{t+1}, \mathcal{A}_t, \mathcal{R}_t)$ in $\mathcal{B}$.
  if *enough experiences in* $\mathcal{B}$ **then** // Learning
    Sample minibatch of M transitions from $\mathcal{B}$.
    for *each* $(\mathcal{S}_m, \mathcal{S}_{m+1}, \mathcal{A}_m, \mathcal{R}_m, done_m) \in M$ **do**
      if $done_m$ **then**
        $y_m \leftarrow \mathcal{R}_m$
      end
      else
        $y_m \leftarrow \mathcal{R}_m + \gamma \max_{\mathcal{A}_{m+1}} \hat{Q}(\mathcal{S}_{m+1}, \mathcal{A}_{m+1})$
      end
    end
    Loss $\mathcal{L} \leftarrow \frac{1}{M} \sum_{m=0}^{M-1} (Q(\mathcal{S}_m, \mathcal{A}_m) - y_m)^2$
    Update Q using SGD by minimizing $\mathcal{L}$.
    Every $\mathcal{C}$ steps, *copy* weights from Q to $\hat{Q}$.
  end
end

V. Results

For the evaluation, we simulate a factory floor, spanning $12m \times 12m$. Four autonomous robots are positioned within the environment, each one tethered with four designated sensors, continuously collecting data from the environment and transmitting them to their connected robot for processing. The available frequency capacity of each robot ranges between $2GHz$ and $5GHz$ and their energy resources between $1500J$ and $6000J$. Task assignments are standardized, with the robot executing a single application characterized by data size of $L = 200KB$ and computational demands of $c = 300Mcycles$. The rest of the system parameters are set as follows: the effective switch capacitance K is set to 10^{-27},

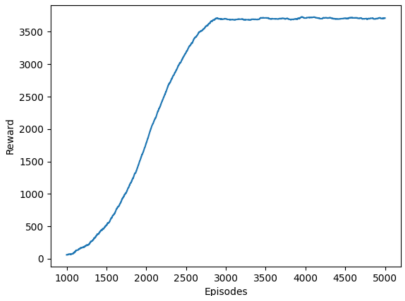
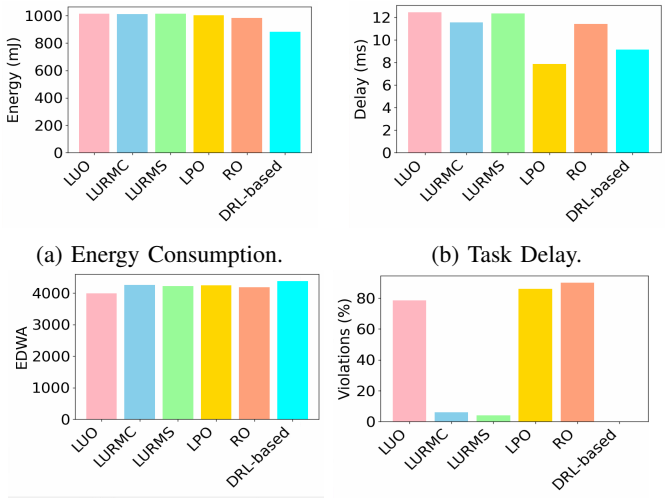


Fig. 2: Training Process Convergence.



(a) Energy Consumption. (b) Task Delay.

(c) EDWA (d) Constraint Violations.

Fig. 3: Benchmarking the proposed mechanism.

the transmission power P for both sensors and robots is set at $20dBm$, bandwidth $b = 1.2GHz$, the channel gain h is set to 10^{-7}, and the noise variance σ^2 at 10^{-12}.

The training phase spans 10^4 episodes. Regarding the training parameters, we fine-tune the learning rate ζ set to 0.03 and the discount factor γ at 0.99. The DNN's architecture comprises two hidden layers with 256 and 512 neurons respectively. The experience replay memory has a size of 5×10^4 while we use a batch size M of 512. The weights of the target network $\hat{Q}$ are iteratively updated at each step, ensuring adaptability and responsiveness to evolving environmental conditions. The convergence plot, illustrated in Fig. 2, provides a comprehensive overview of this iterative learning process, depicting the model's progression towards increasingly effective decision-making. Here, we showcase a moving average of the last 100 values of the total system reward. We see that the agent learns the optimal policy rather quickly, in less than 3000 episodes.

We benchmark our mechanism against the following five baselines: i) the *Least Used Offloading (LUO)* algorithm which greedily prioritizes task offloading to robots with the most available resources, thus optimizing resource utilization and ensuring balanced task distribution across the system.

The mobility here is randomly decided. ii) the *Least Used & Robot Movement towards the Sensors (LURMS)* algorithm, which builds upon LUO but also incentivizes the robots movements towards the sensors, enabling robots for efficient data collection, thereby minimizing sensor-to-robots transmission delays. iii) the *Least Used & Robots movement towards Clustering (LURMC)* algorithm, which builds upon LUO but also incentivizes the robot's movements towards the robot it offloads to, optimizing inter-robot communication. iv) the *Local Processing Only (LPO)* algorithm which restricts task processing exclusively on the robot. The robots here are static. v) the *Random Offloading (RO)* algorithm which follows a randomized approach to task allocation and mobility.

In Fig. 3, we present the average results of 100 experiment repetitions initiating from 10 distinct states. The initial environmental conditions were hand-picked to reflect real-world dynamics such as fluctuations in energy availability, computational resources, and bandwidth constraints. We first evaluate the average energy consumption in Fig. 3a. Our mechanism resulted in the most energy efficient trajectory planning and sensor task scheduling; this verifies its ability in adapting to the dynamic environment conditions and selecting the optimal movement and offloading actions. Then, we examine the end-to-end average delay performance of all six algorithms in Fig. 3b; here, the LPO yielded the minimum average delay, since all the tasks are executed locally without additional consideration of energy consumption. The proposed mechanism consistently outperformed the rest of the baseline algorithms in minimizing delays, showcasing its ability to optimize task allocation and robot mobility while keeping the energy consumption low.

The first two performance evaluations give more insight about the superiority of our mechanism independently for the two main metrics under consideration. Additionally, in Fig. 3c, we depict the *Energy-Delay Weighted Average (EDWA)* score for all algorithms. This metric is the average of the objective function (9). Here, our algorithm clearly outperforms all the others. This solidifies that our approach has a good ability of finding a balance between energy consumption and delay minimization in a complex, dynamic environment like the one envisioned. Finally, in Fig. 3d we demonstrate the reliability performance of each alternative, in the form of energy and delay constraint violations. Here, we notice that the static, greedy solutions, i.e., LU and LPO, as well as the random one, RO, result in the most constraint violations per 100 experiments ($\approx 80\%$). The proposed algorithm achieved a perfect reliability score across all scenarios, showcasing its adaptive nature and robust decision-making capabilities. As a close second and third, LURMS and LURMC managed well to respect the thresholds showing the importance of optimizing the trajectory together with the offloading decisions.

VI. Conclusion

In this paper, we introduced a DRL-based trajectory planning and task scheduling mechanism for edge robotics. We first mathematically formulated the problem of decision-making regarding robot movement and task offloading. Since this problem is very challenging to solve in real-time, we reformulated it as a Markov Decision Process and proposed a DRL-based mechanism to solve it online. Our algorithm showed superior performance against other baselines by minimizing end-to-end task delays and improving energy utilization whilst respecting the system's constraints. Future plans involve fine-tuning the algorithm parameters and exploring alternative RL algorithms for enhancing scalability.

References

[1] F. Saeik *et al.*, "Task offloading in Edge and Cloud Computing: A survey on mathematical, artificial intelligence and control theory solutions", Computer Networks, vol. 195, p. 108177, 2021.

[2] D. Dechouniotis, D. Spatharakis and S. Papavassiliou, "Edge Robotics Experimentation over Next Generation IIoT Testbeds," IEEE/IFIP Network Operations and Management Symposium, 2022, pp. 1-3.

[3] S. R. Behera, N. Panigrahi and S. K. Bhoi, "A Novel Dynamic D2D-assisted Offloading Decision Making in Multi-user Edge Computing Environment," OITS International Conference on Information Technology (OCIT), 2023, pp. 144-149.

[4] M. Avgeris, M. Mechennef, A. Leivadeas, and I. Lambadaris, "A two-stage cooperative reinforcement learning scheme for energy-aware computational offloading", in IEEE 24th Int. Conference on High Performance Switching and Routing (HPSR), 2023, pp. 179–184.

[5] L. A. Grieco, G. Boggia, G. Piro, Y. Jararweh, and C. Campolo (Eds.), "Ad-Hoc, Mobile, and Wireless Networks: 19th International Conference on Ad-Hoc Networks and Wireless, ADHOC-NOW 2020, Bari, Italy, October 19–21, 2020, Proceedings (Vol. 12338)," Springer Nature.

[6] F. Jiang, F. Wei, J. Wang and X. Liu, "Delay-Aware Energy Minimization Offloading Scheme for Mobile Edge Computing," IEEE/CIC Int. Conference on Communications in China (ICCC), 2020, pp. 717-722.

[7] H. Wang, Z. Lin and T. Lv, "Energy and Delay Minimization of Partial Computing Offloading for D2D-Assisted MEC Systems," IEEE Wireless Communications and Networking Conference (WCNC), 2021, pp. 1-6.

[8] M. Li, T. Chen, J. Zeng, X. Zhou, K. Li and H. Qi, "D2D-Assisted Computation Offloading for Mobile Edge Computing Systems with Energy Harvesting," 20th Int. Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT), 2019, pp. 90-95.

[9] N. Lalwani, V. Mehta and S. N. Merchant, "Efficient Resource Allocation for Crowd-Cloud Assisted D2D Computation Offloading," 2019 16th IEEE Annual Consumer Communications and Networking Conference (CCNC), Las Vegas, NV, USA, 2019, pp. 1-2.

[10] Y. Jeon, H. Baek and S. Pack, "Mobility-Aware Optimal Task Offloading in Distributed Edge Computing," 2021 International Conference on Information Networking (ICOIN), 2021, pp. 65-68.

[11] J. Li, W. Liang, M. Chen and Z. Xu, "Mobility-Aware Dynamic Service Placement in D2D-Assisted MEC Environments," IEEE Wireless Communications and Networking Conference (WCNC), 2021, pp. 1-6.

[12] G. Fragkos, N. Kemp, E. E. Tsiropoulou and S. Papavassiliou, "Artificial Intelligence Empowered UAVs Data Offloading in Mobile Edge Computing," IEEE Int. Conf. on Communications (ICC), 2020, pp. 1-7.

[13] X. Wang, H. Guo, "Mobility-Aware Computation Offloading for Swarm Robotics using Deep Reinforcement Learning," IEEE Annual Consumer Communications & Networking Conf. (CCNC), 2021, pp. 1-4.

[14] A. Hameed, J. Violos, A. Leivadeas, N. Santi, R. Grünblatt and N. Mitton, "Toward QoS Prediction Based on Temporal Transformers for IoT Applications," in IEEE Transactions on Network and Service Management, vol. 19, no. 4, pp. 4010-4027, 2022.

[15] G. Nieto, I. de la Iglesia, U. López-Novoa and C. Perfecto, "Deep Reinforcement Learning-based Task Offloading in MEC for energy and resource-constrained devices," IEEE Int. Mediterranean Conference on Communications and Networking (MeditCom), 2023. pp. 127-132.