SAPIENZA
UNIVERSITÀ DI ROMA

Autonomous Networking a.y. 22-23

Homework 1: Report

Ilaria De Sio - desio.2064970@studenti.uniroma1.it
Paolo Pio Bevilacqua - bevilacqua.2002288@studenti.uniroma1.it
Chiara Ballanti - ballanti.1844613@studenti.uniroma1.it

February 1, 2025

1 Introduction

In this homework we addressed a drone routing problem using a Reinforcement Learning technique. We have a set of N drones $\{d_0, ..., d_{N-1}\}$ deployed in Area of Interest (AoI). During each simulation all the drones can move within the AoI. Each drone moves independently from each other following his path and capturing events. Events generate packets that has to be sent to the depot, that is a fixed sink D deployed in the AoI. Every packet expires after 2000 time-steps (or 300 seconds). We can't modify the trajectory of each drone, so we have to implement an algorithm that allows multi-hop drone-to-depot (node-to-sink) communication. The aim of our algorithm is to maximize the number of packets delivered to the depot on time while minimizing delivery time.
Once the packet is generated, the drone can decide to keep the packet in its buffer, or send the packet to one of its neighbors.
In particular we have to design a Reinforcement Learning algorithm that intelligently guides the drones in a choice.

2 Learning Model

In this section, we describe the problem modelling as Reinforcement Learning problem.

2.1 States representation

Each state is represented by the pair consisting of the direction of the drone and the index of the cell in which it is located:

$$\text{State} = (\text{direction}, \text{cell})$$

1. **Direction of the drone**: We drew a circumference around each drone and divided it into circular sectors. We calculate the trajectory vector of the drone using the coordinates of its current position and its $next\_target()$. The direction is given by the circular sector containing the trajectory vector.
   We have 4 circular sectors, so the value of the direction is a natural number that ranges between 1 and 4.

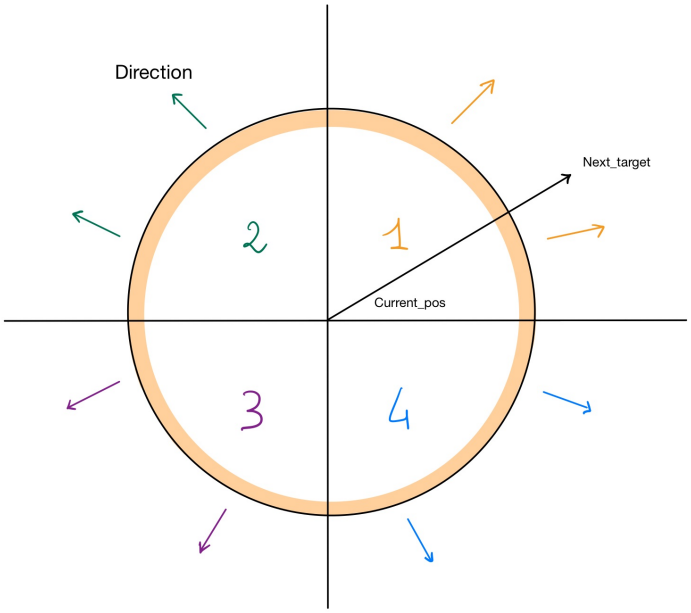


Figure 1: Representation of the direction of a drone

2. **Index of the cell**: We divided the AoI of the simulation into cells. Using the drone's coordinates it is possible to determine the index of the cell in which the drone is located.
   We have 4 cells, so the value of the index is a natural number that ranges between 0 and 3.

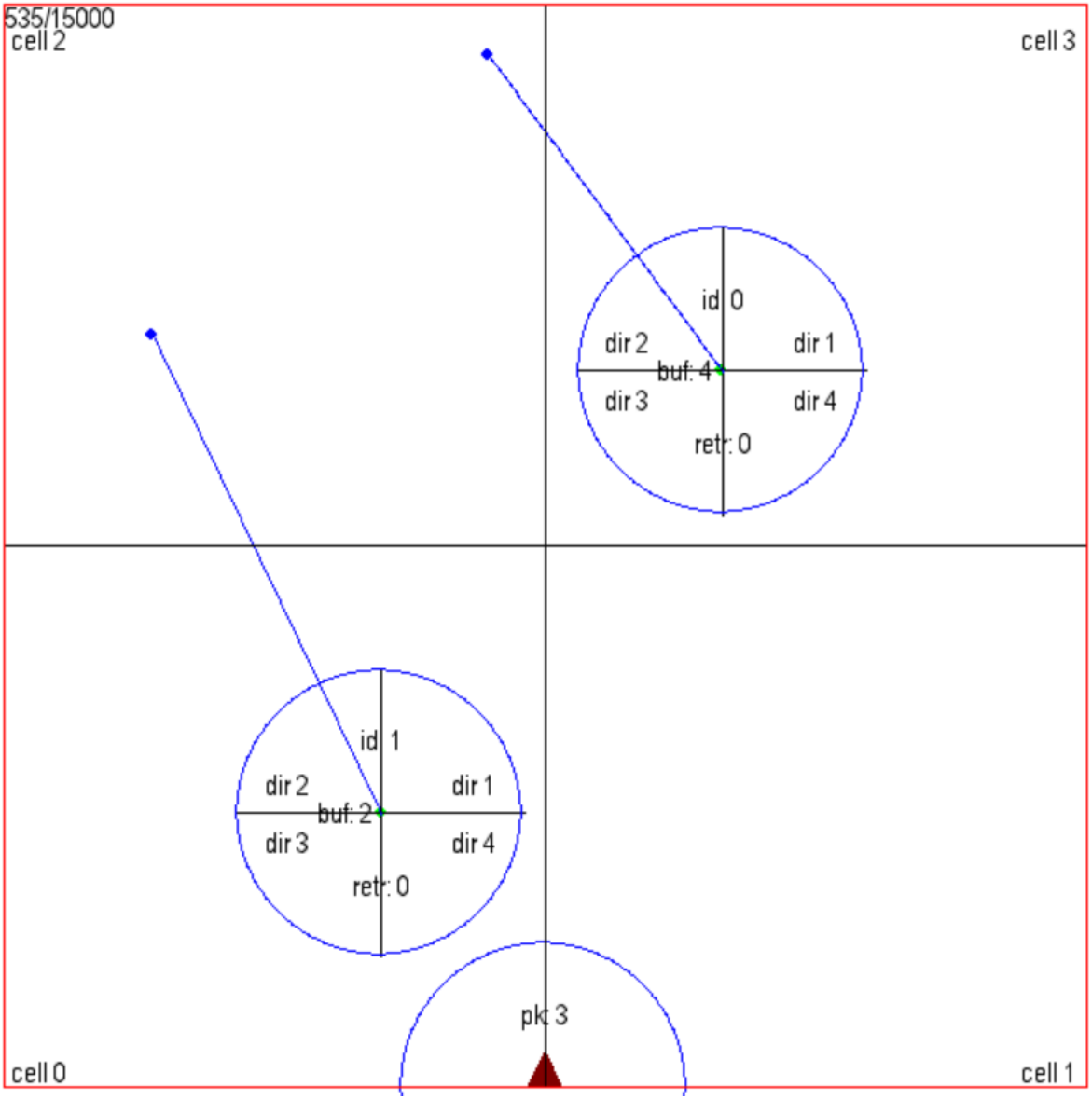


Figure 2: State representation of two drones moving in the AoI during the simulation.

2.2 Actions

In our algorithm an agent (a drone) can perform two possible actions:

1. **Keep**: the drone decides to keep its packets;
2. **Move**: the drone decides to send its packets to one of the neighbors.

2.3 Q-table

Each agent (drone) has its own Q-table which is updated on the basis of what such agent experiences during the simulation.
We used the Optimistic initial value technique to make the drone prefer a certain action over the other one at the beginning.

States	**Actions**	
	Keep	Move
(1, 0)	0	*optimistic_init_value*
(2, 0)	0	*optimistic_init_value*
(3, 0)	0	*optimistic_init_value*
(4, 0)	*optimistic_init_value*	0
(1, 1)	0	*optimistic_init_value*
(2, 1)	0	*optimistic_init_value*
(3, 1)	*optimistic_init_value*	0
(4, 1)	0	*optimistic_init_value*
(1, 2)	0	*optimistic_init_value*
(2, 2)	0	*optimistic_init_value*
(3, 2)	0	*optimistic_init_value*
(4, 2)	*optimistic_init_value*	0
(1, 3)	0	*optimistic_init_value*
(2, 3)	0	*optimistic_init_value*
(3, 3)	*optimistic_init_value*	0
(4, 3)	0	*optimistic_init_value*

Table 1: Initial Q-table

2.4 Relay Selection

The *relay_selection*() function is called by the simulator every time an agent (a drone) has to choose an action to select a relay for packets.
We chose the epsilon-greedy strategy in order to decide if the agent (the drone) should explore or exploit:

$$A_t \leftarrow \begin{cases} \arg\max_a Q(s_t, a) & p = 1 - \epsilon \quad \text{(exploitation)} \\ a \sim \text{Uniform}(\{a_1, ..., a_k\}) & p = \epsilon \quad \text{(exploration)} \end{cases} \tag{1}$$

If the chosen action is $Keep$ the drone will ignore its neighbors to keep the packet. If the chosen action is $Move$, a relay for packets will be selected according to geographic routing using C2S criteria.

In order to record the performed actions we use a data structure called *taken_actions*, described as follows:

$$taken\_actions = \{id\_event : (current\_state,\ action,\ current\_waypoint)\} \tag{2}$$

Note that we will need the *current_waypoint* to calculate the *next_state*, which is required in the Bellman equation (5).

2.5 Feedback

The $feedback()$ function is called by the simulator to give a reward to the drones when a packet expires ($outcome = -1$) or arrives to the depot ($outcome = 1$).
We used the following reward function:

$$reward \leftarrow \begin{cases} \dfrac{(2000 - delay)}{50} & outcome = 1 \\ -20 & outcome = -1 \end{cases} \tag{3}$$

Note that $delay \in [0,\ 2000]$, so $reward \in (0,\ 40]\ \cup\ \{-20\}$.

Once the reward is calculated, it is used to compute the temporal difference and update the Q-table according to the Bellman equation.

The Temporal Difference is calculated as follows:

$$td = reward_{t+1} + \gamma \cdot \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \tag{4}$$

The Bellman equation is the following:

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha \cdot td \tag{5}$$

where α is the learning rate and γ is the discount factor.

3 Experiments

In this section, we discuss the simulation settings and the obtained results.

3.1 Setup

We explain our implementation choices on the values of α, γ and ϵ.

- $\alpha = \begin{cases} 0.8 & 0 <= p < 0.25 \\ 0.6 & 0.25 <= p < 0.50 \\ 0.4 & 0.50 <= p < 0.75 \\ 0.2 & 0.75 <= p <= 1 \end{cases}$ with $p =$ curr_step / simulation_len.

  We decided to set the learning rate variable during the time. At the beginning of the simulation the agent has to learn from the environment. For this reason α will have a high value. While the simulation goes on, α will decrease its value. Therefore, the agent will give less weight to what he learns in the next steps. The rewards obtained from actions will therefore have a decreasing impact.

- $\gamma =$ curr_step / simulation_len:
  We decided to set the discount factor variable during the time. It quantifies how much importance we give for future rewards. At the beginning of the simulation the agent doesn't have information on future states. For this reason γ will have a low value, so the agent will tend to consider only immediate rewards. While the simulation goes on, γ will increase its value, so the agent will consider future rewards with greater weight, willing to delay the reward.

- $\epsilon = 0.05$:
  The agent (the drone) explores with a probability of 0.05.

3.2 Results

In the plots we show the results obtained by our algorithm compared to geographic routing and random routing.
We evaluated our model using the following metrics:

- mean number of relays;
- packet mean delivery time;
- packet mean delivery ratio;
- score = $\sum$ event delays / number of events
  (where expired or not found events will be counted with a $max\_delay \cdot 1.5$).

The model aims to maximize the number of packets delivered to the depot on time while minimizing delivery time.

We tested the performance of our algorithm on every *seed* in order to identify the best.
From the following plot we realized that the best $seed = 0$.

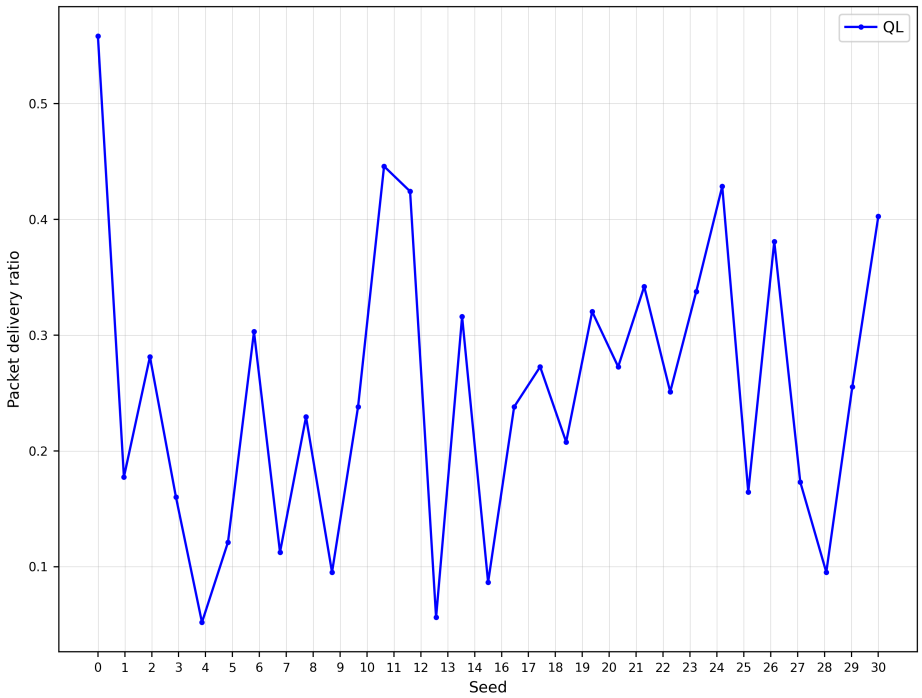
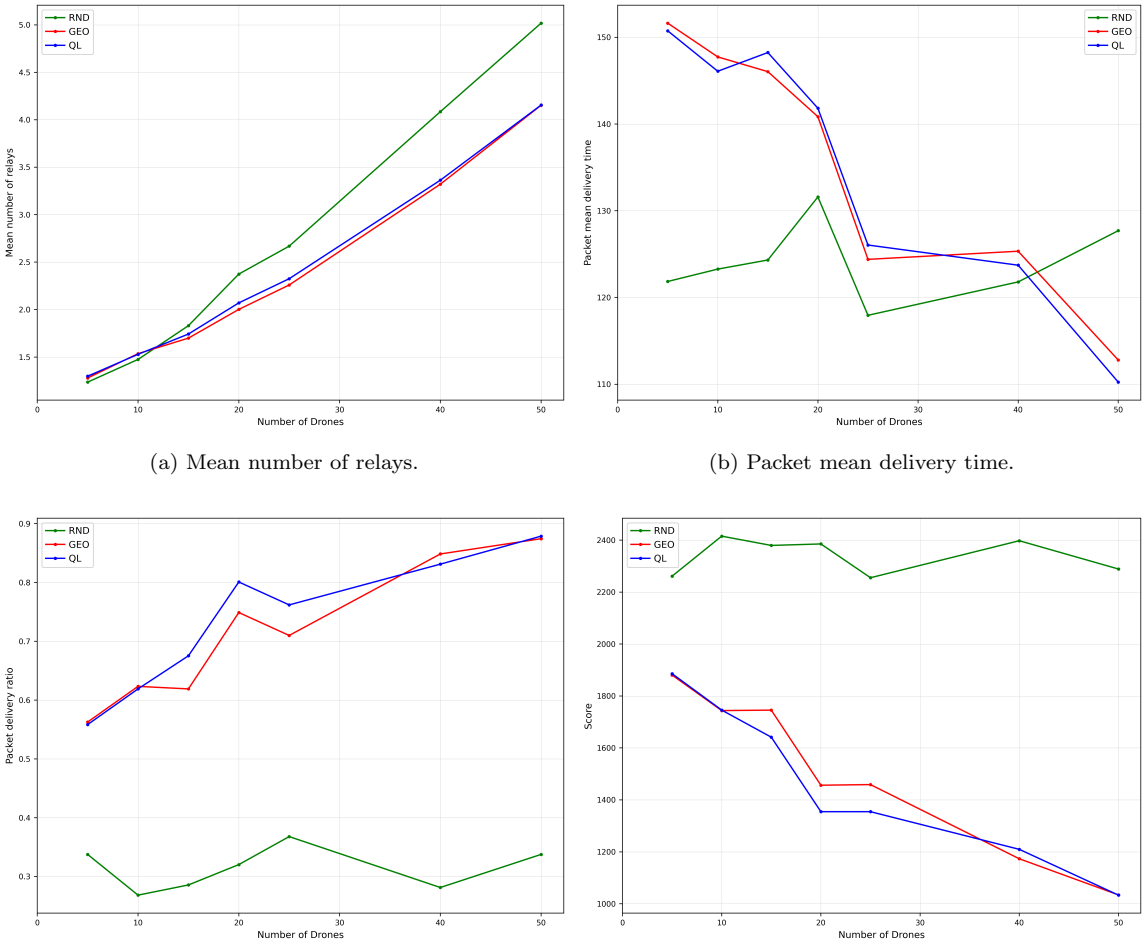


Figure 3: Trend of the packet delivery ratio on 5 drones and 15k simulation duration as the *seed* changes.

We show the results obtained by our approach with different duration of the simulation to understand the behaviour of the model.
The simulation with 15k, 25k and 50k are shown respectively in figure 4, 5 and 6.

Plots on 15k simulation duration :



(a) Mean number of relays.

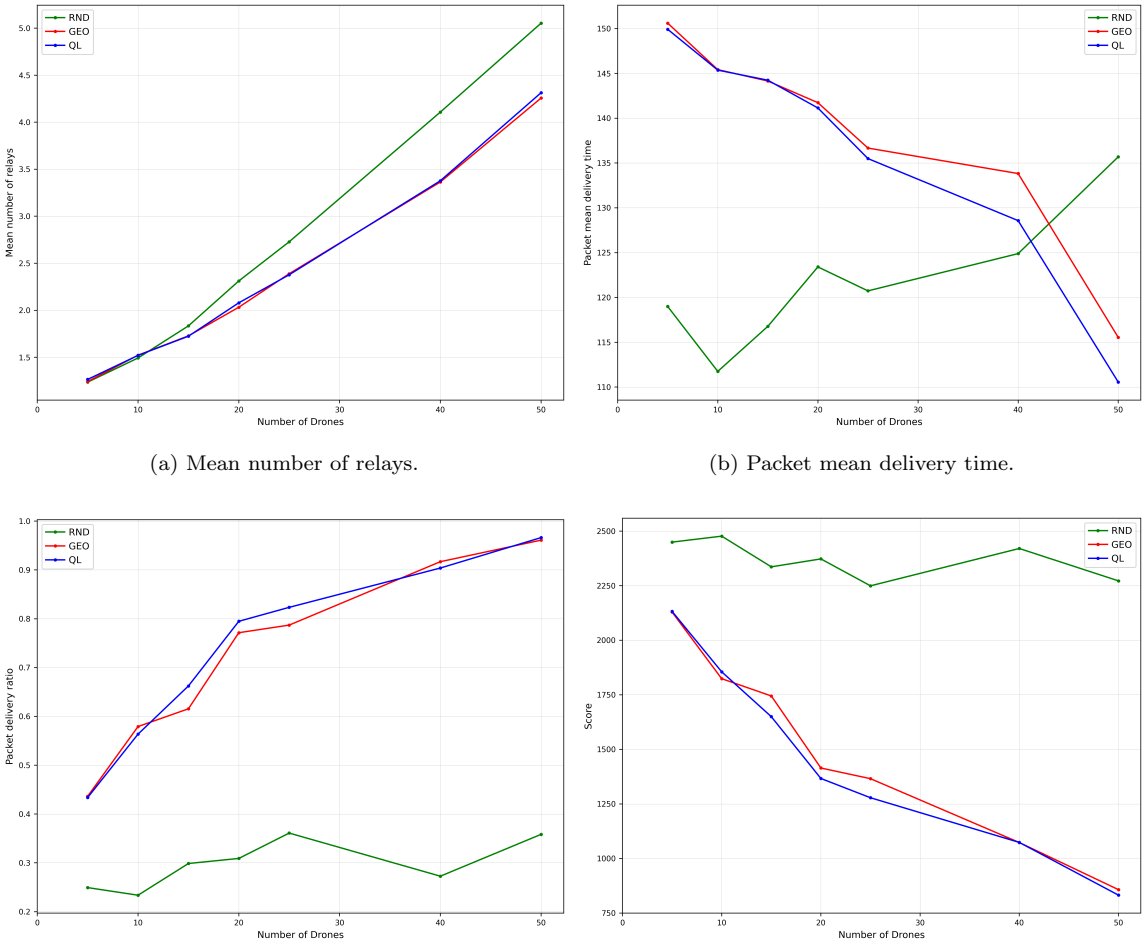
(b) Packet mean delivery time.

(c) Packet delivery ratio.

(d) Score.

Figure 4: Plots on 15k simulation duration for the four metrics.

Plots on 25k simulation duration :



(a) Mean number of relays.

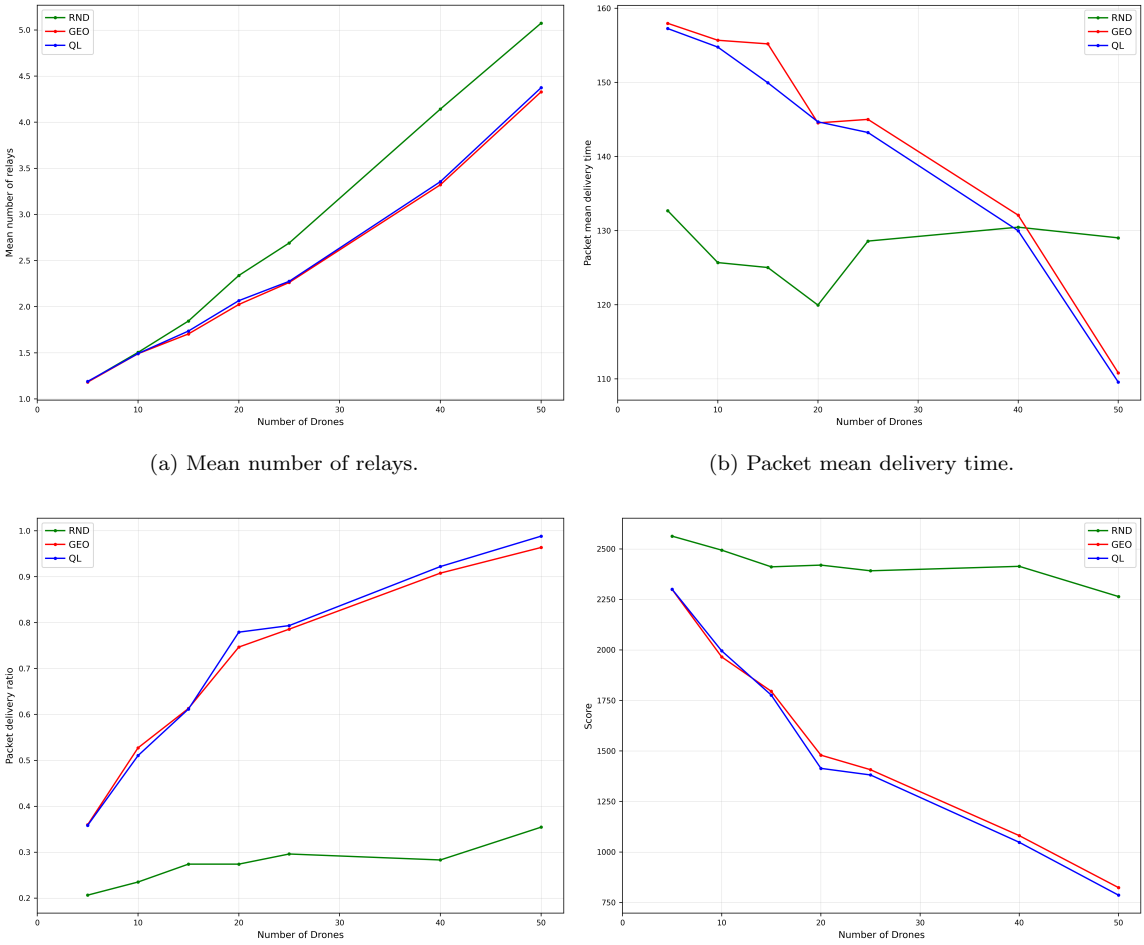
(b) Packet mean delivery time.

(c) Packet delivery ratio.

(d) Score.

Figure 5: Plots on 25k simulation duration for the four metrics.

Plots on 50k simulation duration :



(a) Mean number of relays.

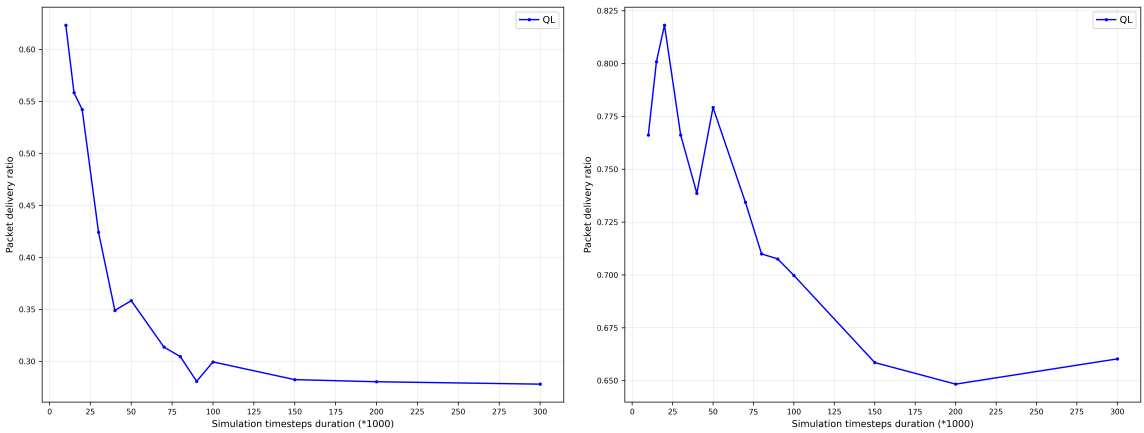
(b) Packet mean delivery time.

(c) Packet delivery ratio.

(d) Score.

Figure 6: Plots on 50k simulation duration for the four metrics.

The following plot shows the trend of packet delivery ratio as the number of time-steps of the simulation changes.
As can be seen, the value will tend to stabilize over time. The model converges, and consequently the reward should also converge to a certain value.



(a) 5 drones (b) 20 drones

Figure 7: Trend of the packet delivery ratio as the simulation duration changes.

4 Conclusions

We noticed that performance varies considerably depending on the number of times the drones enter the communication range of the depot. With a high number of drones performances increased because there is a higher probability that one of them, following its mission, will enter the communication range of the depot. With a low number of drones, there is the possibility that, for a period of time longer than 2000 seconds, no drone enters the communication range of the depot. Packages not delivered before that time will definitely be lost.

As shown in section 3, the QL algorithm outperforms the random routing approach. QL and geographic routing have a similar trend. The two graphs are similar because QL chooses the relay with the geographic routing algorithm if the agent decides not to keep the packets. Despite the strong dependence on geographic routing, QL still achieves better performance.

Contributions

The algorithm's code have been developed in collaborative manner with scheduled meetings on Google Meet platform and using a GitHub repository and the PyCharm's "Code with me" plugin. We discussed all the problems and possible solutions together. Same approach has been used to write the report.

Ilaria De Sio

Paolo Pio Bevilacqua

Chiara Ballanti