# Autonomous Networking a.y. 22-23

## *Homework 1: Report*

Alessio Lucciola - lucciola.1823638@studenti.uniroma1.it
Danilo Corsi - corsi.1742375@studenti.uniroma1.it
Domiziano Scarcelli - scarcelli.1872664@studenti.uniroma1.it

December 3, 2022

## Contents

## 1 Introduction

In this homework, we addressed a drone routing protocol problem using a Q-Learning approach that allows multi-hop drone-to-depot (node-to-sink) communication. We worked on a set of n drones. In each experiment, the number of drones was changed to see what was the difference in terms of performance. Once a drone generates a packet, it can either keep it in the buffer until it reaches the depot or transmits all the packet to another drone. Each packet has an expiration time: when it reaches "2000" then the packet is considered to be lost.
In this homework we were requested to design one or more algorithms that given a drone $\mathrm{d}_0$, its buffered packets $\mathrm{d}_0$, and the drone's neighbors $\mathrm{N}(\mathrm{d}_0)$, eventually selects a neighboring drone $\mathrm{d}_\mathrm{i} \in N(d_0)$ as a relay for all the packets contained in its buffer $\mathrm{b}_0$. We had to implement two functions: *relay_selection* and *feedback*. The former is called at each timestamp of the simulation and here the drone decides whether to keep or transfer the packets depending on the given strategy. The latter is called by each drone every time the packet is lost (in this case the outcome is -1) or the packets are delivered to the depot (in this other case the outcome is 1).
We managed to create some algorithms that rely on Q-Learning and try to solve the problem with increasing complexity using different parameters. Then we compared the performance of our algorithms with the random and geographic approaches using some metrics of interest (section 3). We also discussed other experiments we carried out in order to try to get a higher performance (section 4). Finally we collected all the observations and made a summary of all the results (section 5).

# 2 Learning Model

To solve the routing protocol for drones we had to apply QLearning, a reinforcement learning approach that aims at finding an optimal policy in the sense of maximizing the expected value of the total reward over any and all successive steps, starting from a certain state. In this section we are going to analyze how it works while in the next section we are going to explain how we applied it in the routing algorithms.

## 2.1 Q-learning

Q-learning is a reinforcement learning algorithm that allows the agent to learn the value of an action in a particular state. Q-learning finds an optimal policy $\pi$ that maximizes the total reward.
The reward function $R(a, s)$ is a function that assigns a value for every action $a$ given the agent is in the state $s$. The higher the value, the better the action, and vice-versa. The reward will be used by the agent to update its Q-table. The Q-table is a data structure that contains the current value for every combination of action $a$ and state $s$. Given a certain action $a$ and a state $s$, the formula used to update the Q-table is the following:

$$Q[s][a] = Q[s][a] + \alpha \cdot \Big( r + \gamma \cdot \max\left(Q[n(s)]\right) - Q[s][a] \Big) \tag{1}$$

where $n(s)$ is the next state, r is the reward, $\alpha$ is the learning rate and $\gamma$ is the discount factor that represents the importance of future rewards.

## 2.2 $\epsilon$-greedy

In order to solve the exploration-exploitation dilemma, we decided to use the $\epsilon$-greedy method. This method uses an exploration factor $\epsilon$, which is a number in the interval $[0, 1]$ which describes the probability that the agent has to explore in a certain time step. Our goal was to give a higher probability to explore at the beginning, and then exploit the known information as the Q-table filled up.

## 3 Experiments

We implemented different algorithms, starting with a simple one that uses only the Q-table values to select the best actions and used this algorithm as a starting point to consider other approaches with increasing complexity. The algorithms are the following:

- SimpleQL
- GeoQL

### 3.1 SimpleQL

This is the simplest implementation of a Q-learning-based algorithm that simply relies on the Q-table values. In the *relay_selection* function we perform both exploration and exploitation using the $\epsilon$-greedy approach. At each iteration a random value $rnd_n$ in the interval [0, 1] is chosen and the exploration case is performed accordingly: if $rnd_n \leq \epsilon$ then we choose the exploration case, otherwise we choose the exploitation case. Since we would like to explore more at the first iterations, we set a high $\epsilon$ and every time we perform exploration, $\epsilon$ gets decreased until it reaches 0.1. This means that every time we explore, we reduce the probability to explore at the next iterations.
In the exploration case, the drone selects an action according to this schema:

- Self: If there are no neighbors or if the drone is in the communication range of the depot (meaning that it can immediately send the packet to the depot without transferring them to another neighbor)
- A random drone: Otherwise

In the exploitation case, the drone selects an action according to this schema:

- Self: If there are no neighbors or if the drone is in the communication range of the depot
- The best drone: If there are some neighbors. The best drone among the neighbors is chosen according to the max value in the Q-table

This is how an action is chosen in the *relay_selection* function. About the *feedback* function, we assign a reward according to this function:

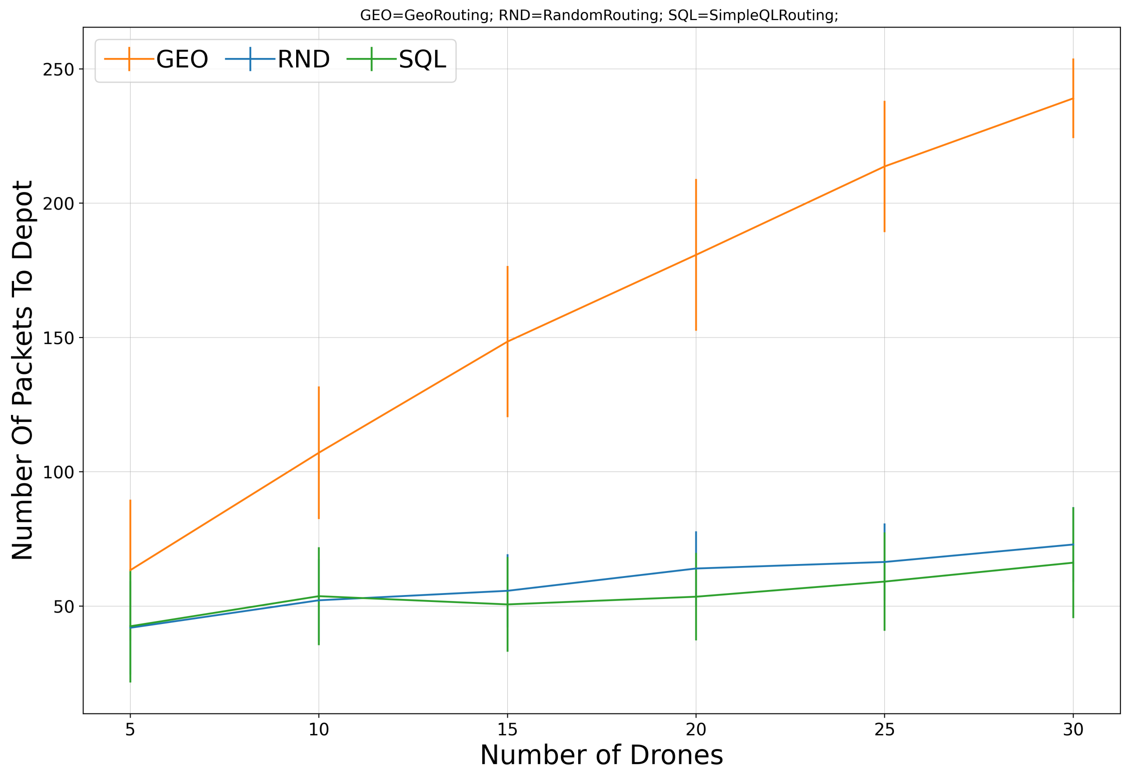
$$r = \begin{cases} -15 & outcome = -1 \land 2000 \leq delay < 3500 \\ -30 & outcome = -1 \land delay \geq 3500 \\ (pkts * 2) + 30 & outcome = 1 \land delay \leq 1000 \\ (pkts * 2) + 15 & outcome = 1 \land 1000 \leq delay < 2000 \end{cases} \tag{2}$$

where *pkts* are the number of packets in the drone buffer. So, if the packets get lost we assign a negative reward depending on the delay (the higher the delay, the larger the negative reward). If the packets are correctly delivered then we assign a positive reward depending on the number of packets

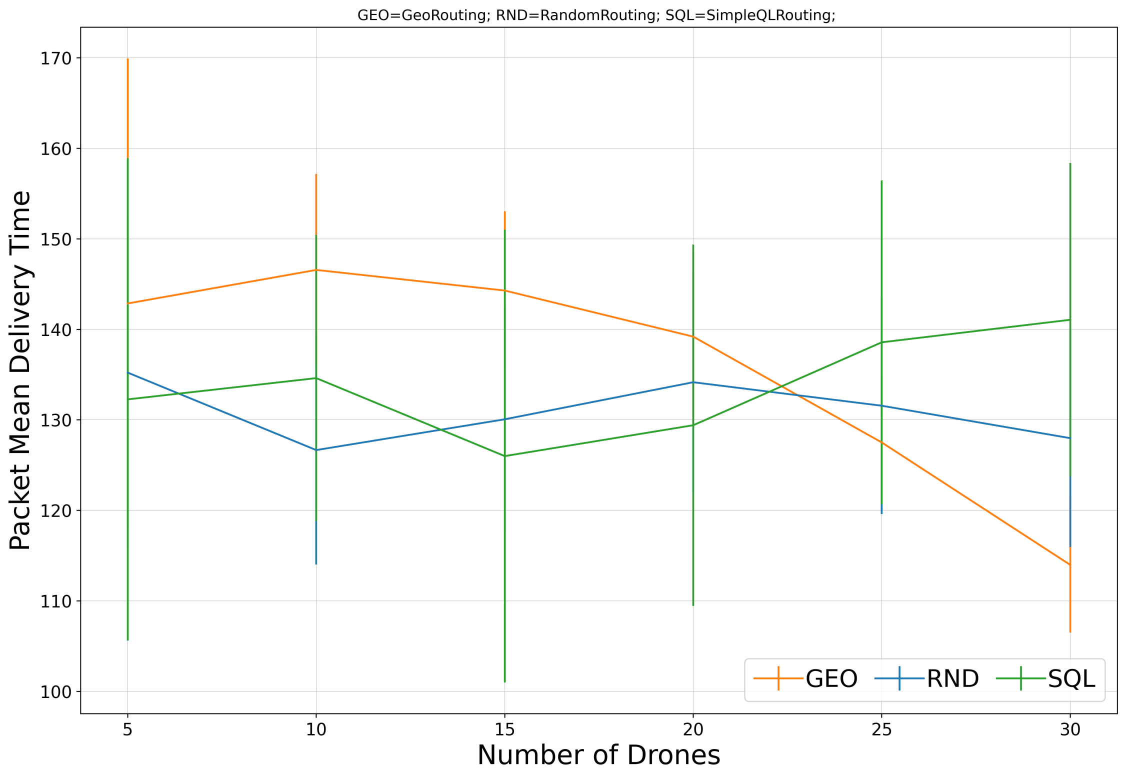
delivered and the delay (the lower the delay, the larger the positive reward).
Finally, we update the Q-table as follows:

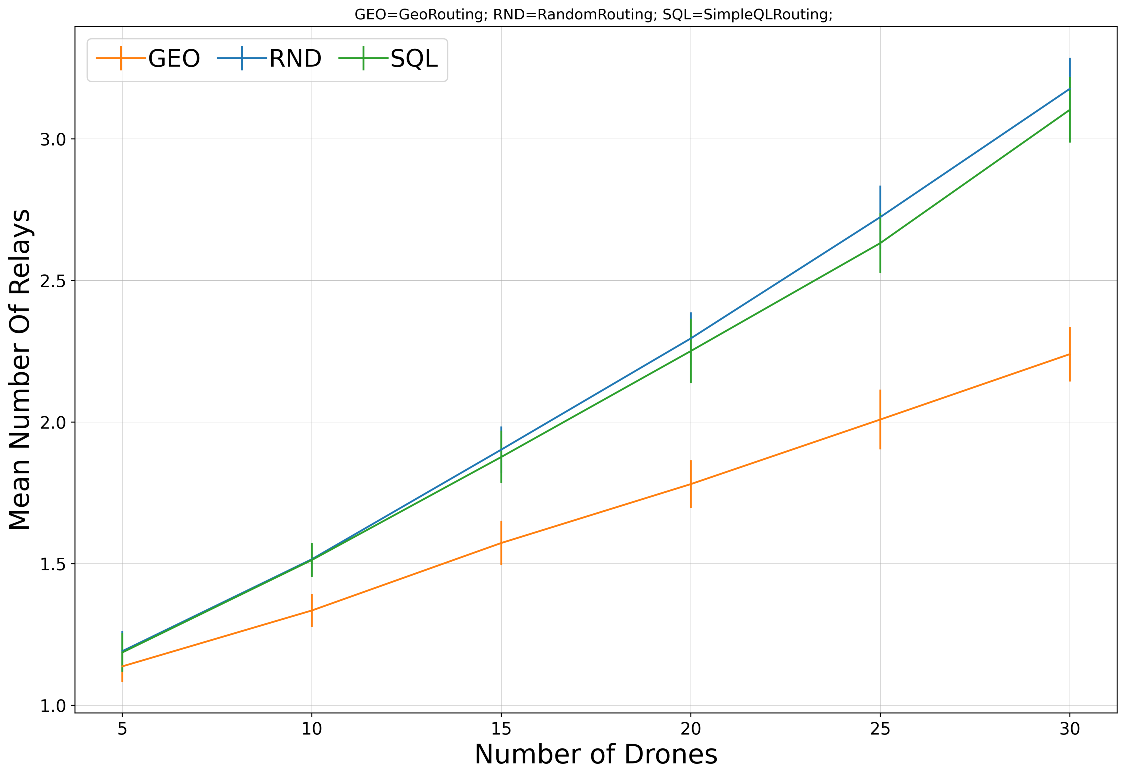
$$Q[c_i][d] = Q[c_i][d] + \alpha \cdot \Big(r + \gamma \cdot \max\left(Q[nc_i]\right) - Q[c_i][d]\Big) \tag{3}$$

where $c_i$ is the cell index (the current state), $nc_i$ is the next cell index (the next state), d is the drone, r is the reward, $\alpha$ is the learning rate and $\gamma$ is the discount factor.
Let's analyze how it performs with respect to the Random Routing and the Geographic Routing algorithm:



It is immediate to notice that the approach that only uses values from the Q-table to select the best action is not enough. The algorithm performs slightly worse then the Random Routing one in which a random action is chosen at each step. Looking at the standard deviation, we can observe how the performance of the SimpleQL algorithm highly depends on the seed. In fact, there are cases in which it overcomes the Random approach and other cases in which it performs worse, but the overall packet delivery ratio for the SimpleQL approach is generally not satisfying for our purpose.

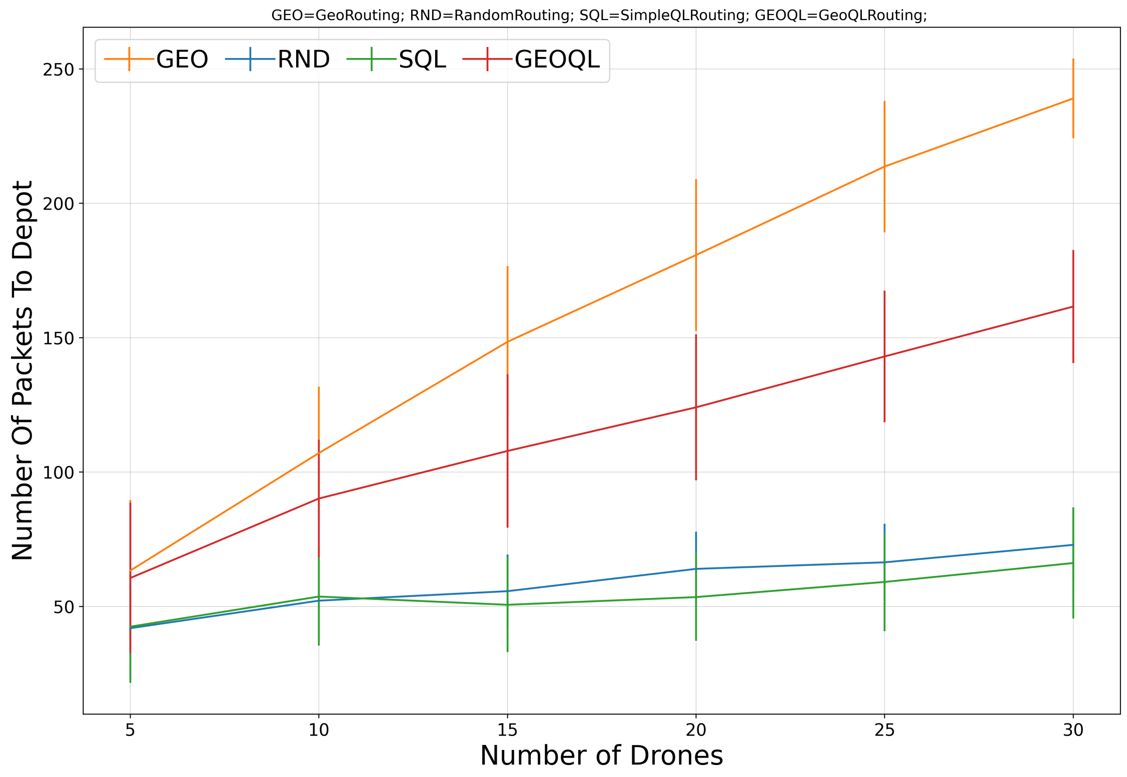

GEO=GeoRouting; RND=RandomRouting; SQL=SimpleQLRouting;
Packet Mean Delivery Time
Number of Drones
GEO
RND
SQL
100
110
120
130
140
150
160
170
5
10
15
20
25
30


GEO=GeoRouting; RND=RandomRouting; SQL=SimpleQLRouting;
Mean Number Of Relays
Number of Drones
GEO
RND
SQL
1.0
1.5
2.0
2.5
3.0
5
10
15
20
25
30

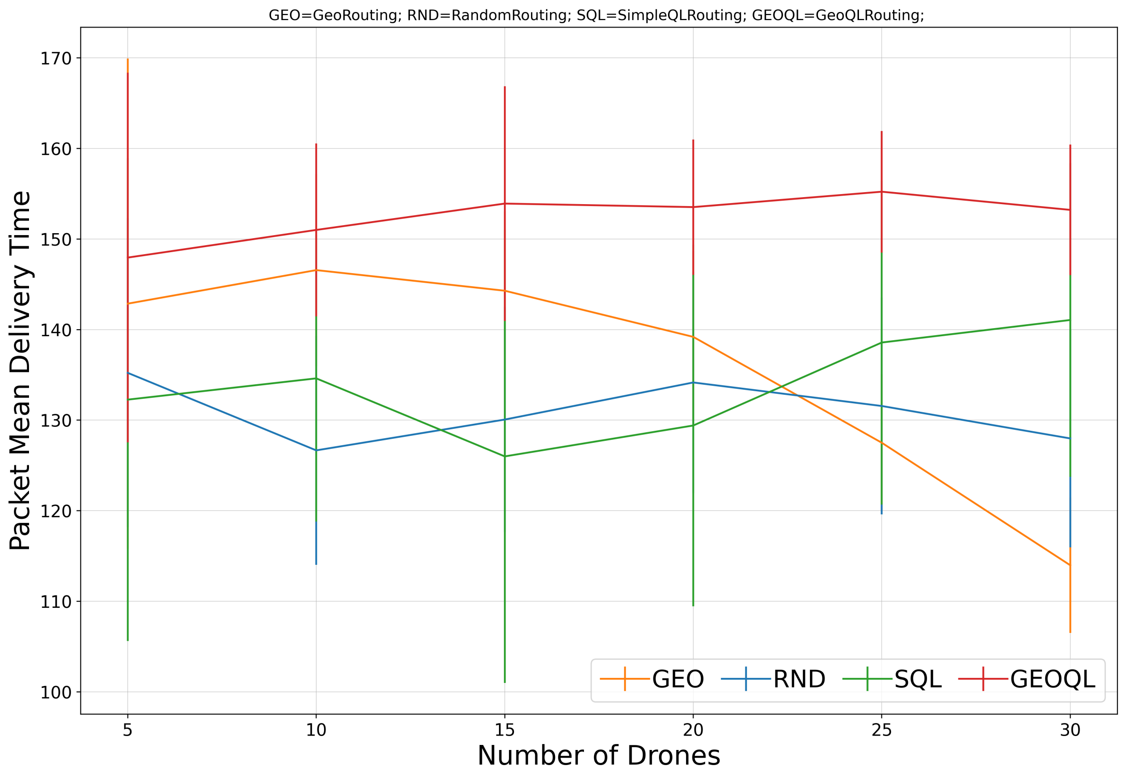
The packet mean delivery time of the SimpleQL algorithm is similar to the Random Routing one and it tends to increase with a larger number of drones. The Geographic Routing algorithm has a higher mean delivery time with a lower number of drones but it decreases a lot with a high population of drones.
Also, the Geographic approach uses a lower number of relays with respect to the SimpleQL and the Random Routing approaches which have almost the same trend, meaning that the former make smarter choices when selecting the best action.
From this result, we understand that the geographical information has to be exploited in order to have an algorithm that has a good packet-to-depot ratio, as well as a lower mean packet delivery time.

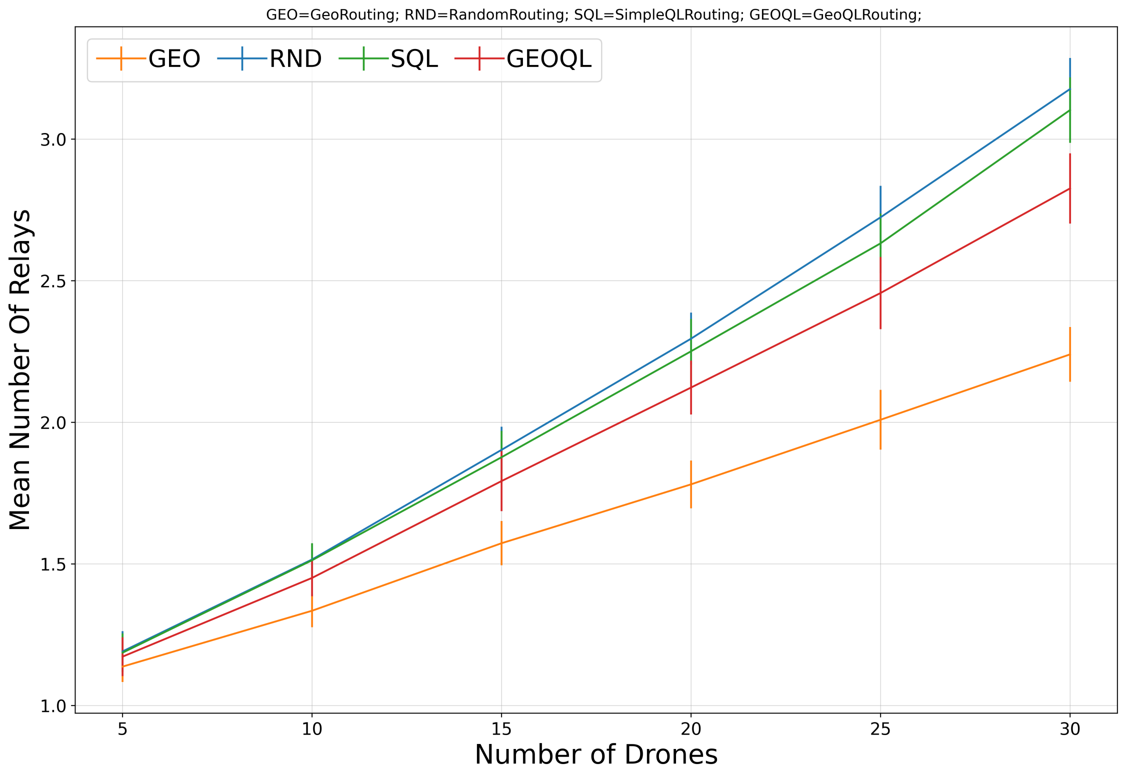
### 3.2 GeoQL

Another implementation is the GeoQL which is a direct enhancement of the SimpleQL algorithm. After having tested the GeoRouting algorithm, we noticed that the performance was higher compared to the Random algorithm and the SimpleQL one. For this reason, we tried to add some geographical information into the simple algorithm and see if we could increase the overall performance. The idea was to exploit the direction of drones to filter which could be the ones to be selected as the action. In particular, we take the current position of each drone and subtract the next target. Basically, if the drone is moving toward the depot, the result will be a positive number, if it is moving in the opposite direction the result will be negative. Finally, we select the subset of drones that have a positive result and choose the one with the highest value in the Q-table as the action. It might happen that in a certain iteration, every drone is moving in the opposite direction of the depot (i.e. the subtractions between the current position and the next target are negative numbers): in this case, the drone itself is selected as the action. This is the exploitation case.
The exploration case, as well as the reward function, are the same as the SimpleQL algorithm.
To sum up, we use the direction of drones to select which ones are heading to the depot, and from this pool of drones we select the one with the highest value in the Q-table.



As we can see, exploiting the geographical information improved the packet-to-depot ratio. Every time we need to select an action, we use the direction of the drones to filter which are the drones from which to select the values of the Q-table. Since the direction is a parameter that gives good results, it is directly used to update the Q-table depending on the outcome of the previous actions. The more information we gain, the better the chosen action is.
The packet delivery ratio is still not as good as the complete Geographic approach, primarily with a high number of drones. With a lower number of drones they perform mostly the same.


GEO=GeoRouting; RND=RandomRouting; SQL=SimpleQLRouting; GEOQL=GeoQLRouting;
Packet Mean Delivery Time
Number of Drones
GEO
RND
SQL
GEOQL
100
110
120
130
140
150
160
170
5
10
15
20
25
30


GEO=GeoRouting; RND=RandomRouting; SQL=SimpleQLRouting; GEOQL=GeoQLRouting;
Mean Number Of Relays
Number of Drones
GEO
RND
SQL
GEOQL
1.0
1.5
2.0
2.5
3.0
5
10
15
20
25
30

Unfortunately, the mean packet delivery time is higher than the one of all the other approaches. It stays constant regardless of the number of drones and it is higher than the complete Geographic Routing approach especially with a high number of drones.
The mean number of relays is lower than the Random Routing and the SimpleQL Routing approaches but still not as good as the Geographic approach.

### 3.2.1 Hyperparameter tuning

A big part of the experimentation part was to change the hyperparameters learning rate $\alpha$, exploration factor $\epsilon$, and discount factor $\gamma$ in order to achieve the best results possible. After running various tests with multiple combinations of these values, we have seen that the combination that gives the best results, given our system and reward function, is:

$$\alpha = 0.8 \ \gamma = 0.1 \ \epsilon = 0.4 \tag{4}$$

The discount factor $\gamma$ is set to a low value since we're in the contest of a non-stationary network, and so we don't want to give large weights to the future values. On the other hand, the learning rate is higher, since in non-stationary networks it's better for the node to learn fast because of their high mobility. Note that, as said before, the $\epsilon$ factor written above is the starting value, and it will decrease for every exploration until it reaches the value 0.1.

## 4 Other experiments

In this section, we will talk about the other algorithms, techniques, and ideas that we tried to implement, but didn't include in the final version since the performances weren't optimal.

### 4.1 Optimistic initial values vs $\epsilon$-greedy

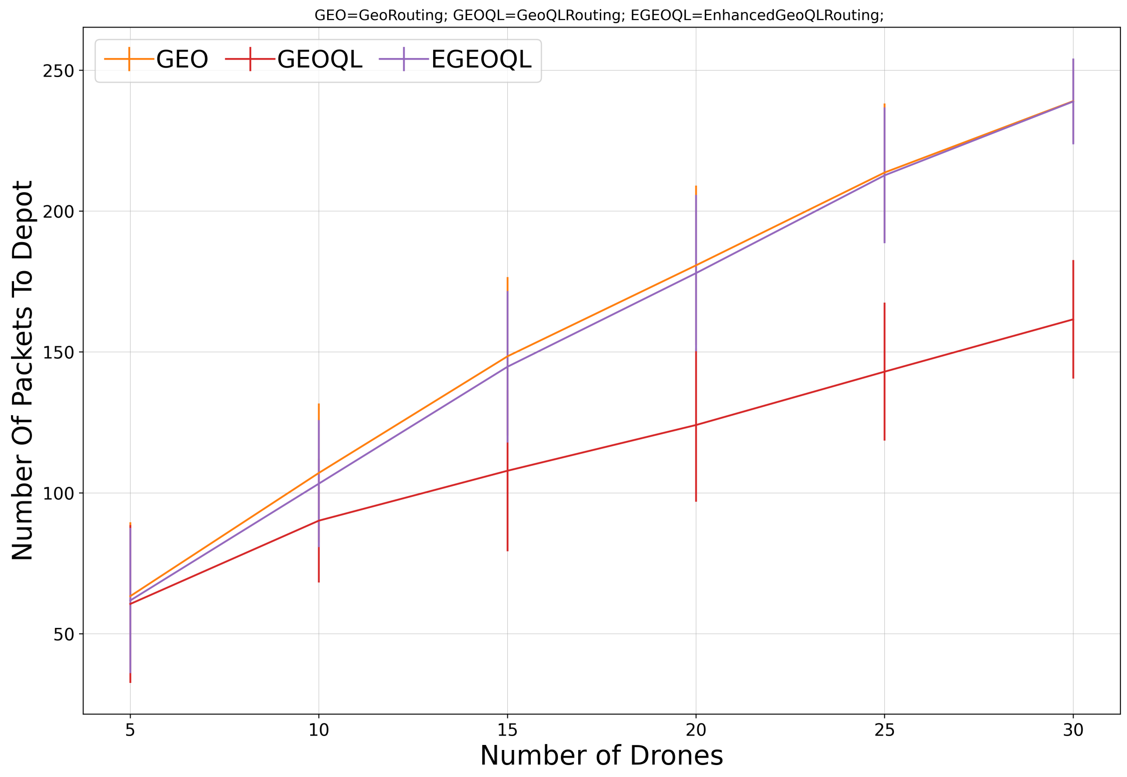
We tried to implement the Optimistic Initial Values approach but it was soon rejected because, according to the theory, it is most effective in stationary problems, exploring only at the beginning. Moreover, being this a problem where drones move continuously, it would hardly have been useful for our purpose.

### 4.2 EnhancedGeoQL

We have tried to implement a Q-learning protocol that would use the same information about the distance between the drone and the depot that is used in the Geographic protocol. The idea was to have a score for every drone that would tell how good it was to pick that specific drone as an action given a certain state. This score was computed with some linear combination between the euclidean distance from the drone to the depot, and the Q value associated with the drone in that particular state. In the exploiting part, instead of taking the drone with the maximum Q value in the current state, we would take the drone with the maximum score.
For example, we tried to compute the array of scores $S$ for each drone with the following formula:

$$S = \frac{1}{4}D_{norm} + \frac{3}{4}Q_{norm} \tag{5}$$

Where $D_{norm}$ is the normalized array that contains the distance from every drone to the depot and $Q_{norm}$ is the normalized array that contains the Q values of every drone in that particular state.
The problem with this approach is that when we computed the score in the exploitation phase, the distance was the value that influenced it the most, even if it was given a lower weight. In the end, the approach was quite almost geographic, and the Q table didn't make the protocol perform better than we hoped.



## 4.3 Better reward function

Since the initial reward function wasn't considering many parameters in order to assign a reward to the drone, we experimented with some more complex functions that also consider some other factors such as the packet delivery ratio at the moment of the reward calculation, the current throughput and the drones' buffer emptiness ratio.

We tested some reward functions inside the GeoQL algorithm, but none of those gave better performances.
An example of a tested reward function:

$$r = \begin{cases} -\frac{d}{1000} + \text{outcome} + 10 \cdot \text{deliveryRatio} & \text{if outcome} = 1 \\ -\frac{d}{1000} + \text{outcome} - 2 \cdot (1 - \text{deliveryRatio}) & \text{if outcome = -1} \end{cases} \tag{6}$$

Where:

- $d \in [0, 2000]$ is the delay;
- deliveryRatio $\in [0, 1]$ is the packet delivery ratio at that moment

## 5 Conclusions

Following all the experiments carried out we can say that a big part of the experimentation was to change the **hyperparameters** learning rate $\alpha$, exploration factor $\epsilon$, and discount factor $\gamma$ in order to achieve the best results possible. Furthermore, we found out that simply using the values of the Q-table to select the best drone (as done in the **SimpleQL** algorithm) is not enough to guarantee acceptable results. Because of that, we understand that the geographical information of the drones has to be exploited in order to have an algorithm that has at least a higher packet-to-depot ratio. In fact, by using the direction of drones to select which ones are heading to the depot, and from these select those with the highest value in the Q-table (as done in the **GeoQL** algorithm), we get an improved packet-to-depot ratio. Unfortunately, the packet delivery ratio is still not as good as the complete Geographic approach, primarily with a high number of drones. With a lower number of drones they perform mostly the same. Also, it seems that a QLearning approach is computationally heavier, as a matter of fact the mean packet delivery time is higher than the one of all the other approaches: it stays constant regardless of the number of drones and it is higher than the complete Geographic Routing approach especially with a high number of drones. The mean number of relays is lower than the Random Routing and the SimpleQL Routing approaches but still not as good as the Geographic approach.

## Contributions

**Alessio Lucciola** SimpleQL, GeoQL, Plotting and Data Elaboration Functions, Report

**Danilo Corsi** SimpleQL and GeoQL (improvements and experiments), Report

**Domiziano Scarcelli** First implementations of SimpleQL, Experiments with EnhancedGeoQL and better reward function, Report