Homework 2 - Autonomous Networking

Andrea Bacciu
Giuseppe Masi
Rocco Pisciuneri

December 2020

Contents

1 Introduction

In this work we address the RL Routing problem, having a single drone (called drone zero) able to sends packets to other drones, called squad drones, which the goal is to send the maximum quantity of packets with minimum delay to the depot. Each drone moves independently from each other following his path, the problem is stationary so the path does not change as shown in figure 1. Each packet is identified by an *event_id*, the drone zero is the only one which implements the mac protocol and use the *relay_selection* to submit an action to perform. The simulation includes some real constraint such as the possibility of transmitting failure from the drone zero to one in the squad and vice-versa in which the drone sends back an acknowledge to the drone zero. The drone zero, given an *event_id* call the *relay_selection* until receive an ACK for the event or until the *event_max_retrasmission* is not reached.

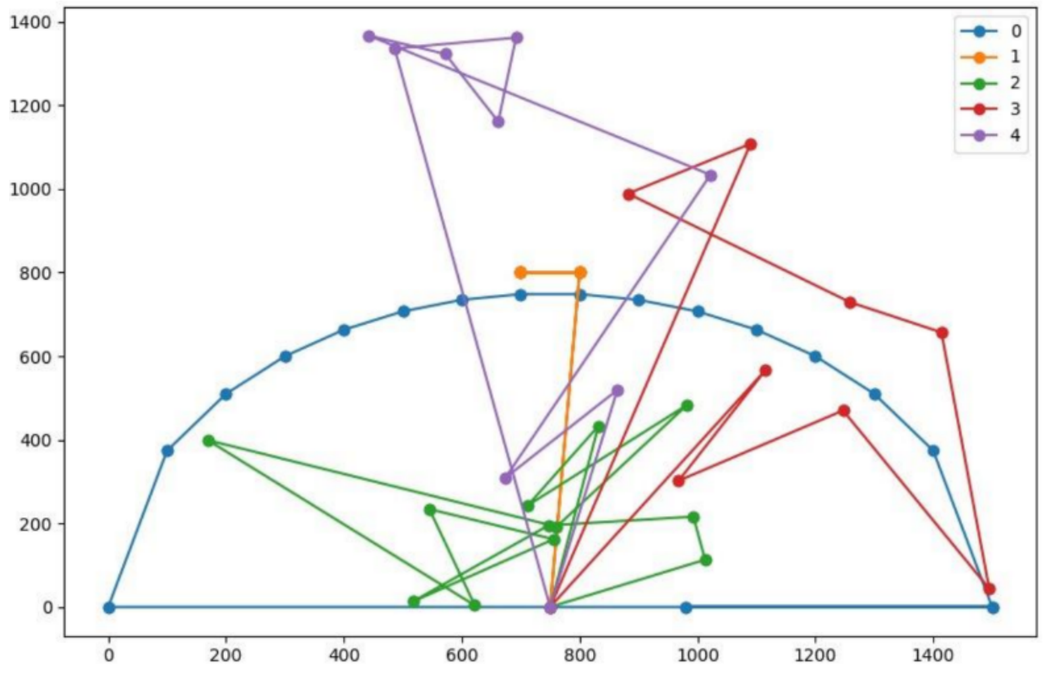

Figure 1: Patrolling scenario with five drones

2 Problem modelling

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

2.1 The state representation

The state is represented by the pair of cell and the set of neighbours of the drone zero. The cell is calculated starting from a division of the action area of the simulation in cells, using the drone zero's coordinates we are able to recognize in which cell it is [1] as shown in figure 2. In each cell, the set of neighbours is not static (because the drones are always in movement), so we have to consider it to perform the best action.

2.2 Actions

An action is represented by the drone id selected by the routing algorithm of the drone zero. The type of actions that can be performed are the following:

1. No transmission: Drone zero decide to keep the packet.
2. Transmission to Drone x, where x in the set of neighbours.

[1] a cell is a discrete number

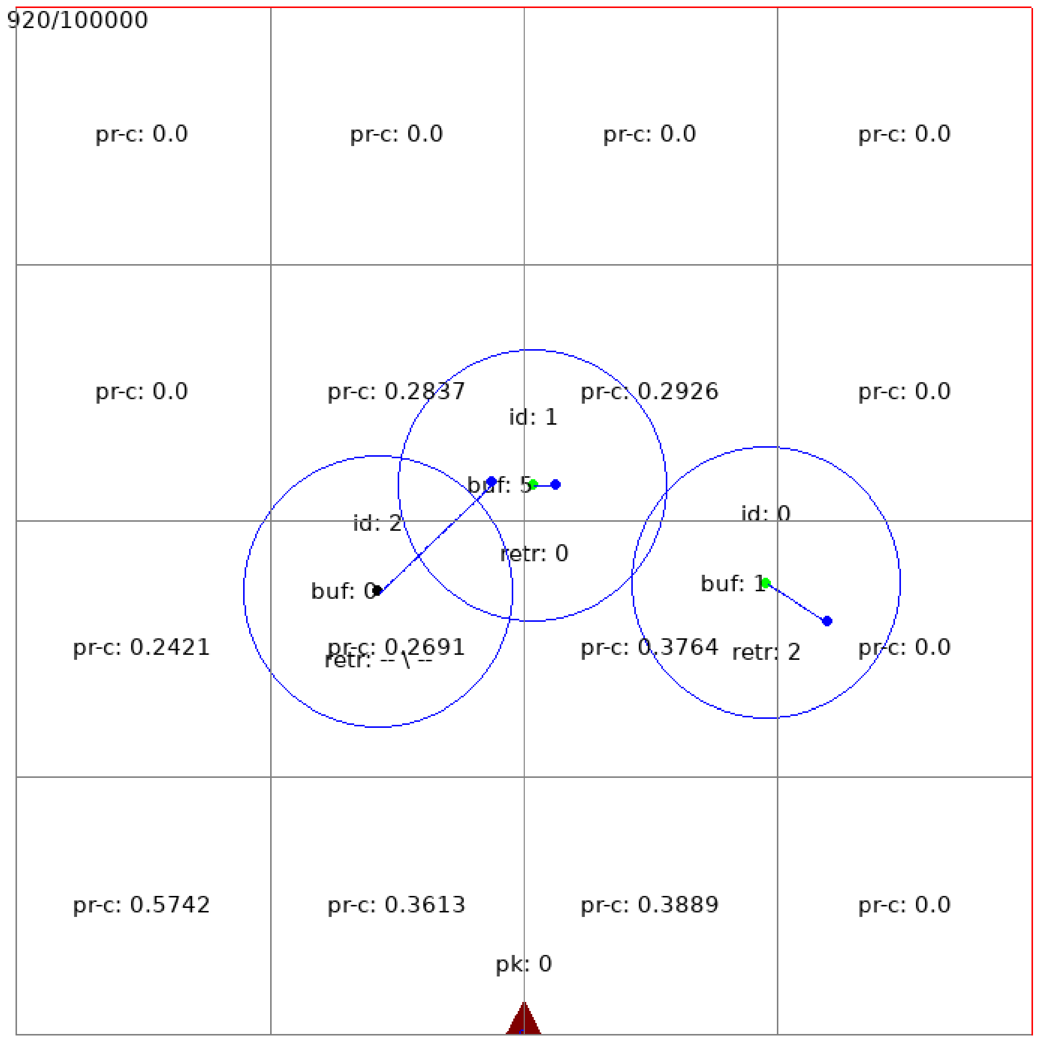

Figure 2: Cells, a cell is a square region in figure

3 Relay Selection

The relay selection api is called by the simulator for a given event and it requires to the drone zero's algorithm to perform an action. In this api we implemented our approach which made use of several data structures and the approach is separated in exploration and exploitation, following the ε-greedy strategy.

To record the action performed we use a data structure called already_taken_action — a.t.a which is

$$a.t.a = \{id\_event : \{state : set(\text{action performed})\}\}$$

a.t.a record the set of all actions performed in a state for a given event. This structure is important because it lets the model update correctly the Q-table having all the necessary information. Also, it will be used in the exploration step to avoid taking the same action multiple times for an event.

3.1 Exploration

The exploration step is performed in two cases:

- **No knowledge on the state:** The exploration is done when the model has no knowledge about the cell, so we have to explore it in order to discover information about the environment.

- **ε-greedy without repetition:** The latter is performed in according to ε-random variable and if the event is already in a.t.a. The exploration step made use of the a.t.a which give us the bijection between the state and the performed actions in that cell for a given event. For each event in the exploration step we select a drone to explore in the set of neighbour minus the set of drones already explored, which is recorded in a.t.a.

  $$\text{drones to explore} = \{\text{neighbours}\} - \{\text{already explored}\}$$

  If the set of drones_to_explore is empty we perform the *No transmission* action. We take at most min(number of actual neighbours, relay selection call) actions randomly avoiding starvation. In the case in which the relay selection call for a particular event is grater than neighbours, we perform the *No transmission* action, because we already explored all neighbours unless there is interference. Once we select the drone to explore we update the a.t.a.

3.2 Exploitation

The exploitation step is performed only if the model has knowledge about the state. The exploration made use of Q-table to select the best drone. As we stated in the section 2.1 the set of neighbours in a cell is not static due to the fact they are always in movement. So accordingly to the Q-table we select the best drone in the cell w.r.t the available neighbours, discarding the unsuccessful drones[2]. The exploitation has priority over the exploration if there's knowledge on the state, and it's performed as first for each new event in order to deliver the packet immediately to the best drone in the cell.

3.3 Problem Faced

We take care of the following aspects:

- **Minimum quantity of packets:** Finding a relay selection strategy limiting the number of sent packets.
- **Manage event in multiple cell:** To update correctly the Q-table in the case in which one *id_event* is sent in multiple cells, we manage this case using the a.t.a data structure. Recording the timestamp in which the drone is selected in a particular cell, we gave priority to the first one, to select it, in another round, as first as possible.

4 Our approaches

In this section we describe two different approaches to the problem which share the state definition and relay selection strategy, they differ on the updating rule and the definition of the Q-table.

4.1 Q-Learning

As first we try an approach inspired by the Q-Learning formulation. The Q-table is used by the exploitation step to select the best drone. In this formulation, we find as the best drone the one in the state with the maximum value in the table. Where α is set to 0.5 after a hyperparameter tuning phase. The formula used is the following:

$$Q_{cell,drone} = Q_{cell,drone} + \alpha(Reward - Q_{cell,drone})$$

[2]Which never obtain a successful outcome

To bring us back to the original formulation we have converted the delay minimization problem into one of reward maximization, so:

$$Reward = events\_duration - delay$$

After a tuning phase we found the ε-value = 2%

4.2 AI

After the Q-learning approach, we move to another experimental approach using always the Q-table, in a minimization setting. The Q-table is initialized with infinite value and we update it only if a drone in a state obtain a lower delay. The updating rule is the following:

$$Q_{cell,drone} = \begin{cases} delay, & \text{if } Q_{cell,drone} > delay \\ Q_{cell,drone}, & \text{otherwise} \end{cases}$$

After a tuning phase we found the ε-value = 2% as shown in figure 6

5 Benchmark & Experimental Setup

In this section, we show the results obtained by our approaches with different duration of the simulation to understand the behaviour of the model. All experiments in this section, use hyper-parameters already tuned for each algorithm in the way shown in the section 6. All plots are produced using seed from 1 up to 30. The model objective is to minimize the following formula:

$$\text{score} = 1.5 * |\text{expired\_packets}| * ttl + \sum_{pck\ \in\ delivered} \text{delivery\_time}$$

The default simulation with 18k, 50k and 100k are shown respectively in figure 3, 4, 5.

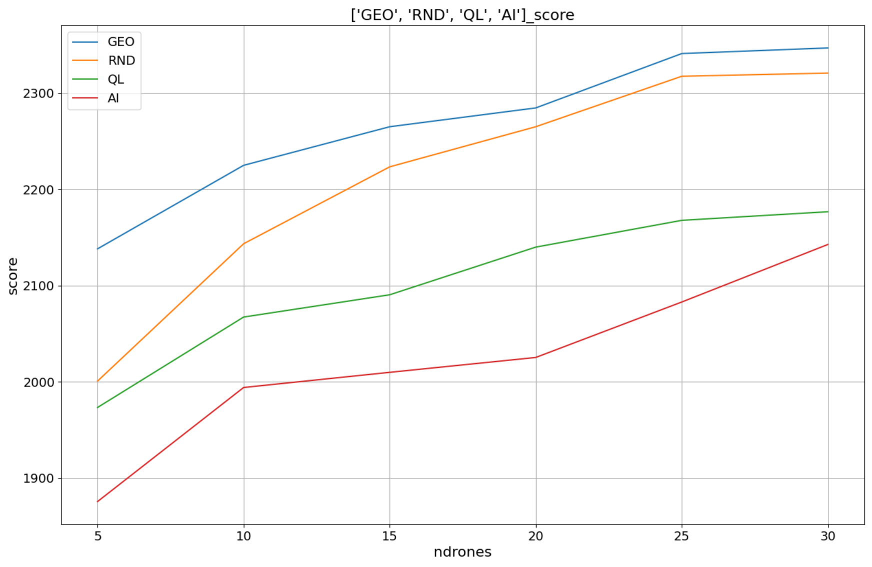

Figure 3: 18k simulation duration - score

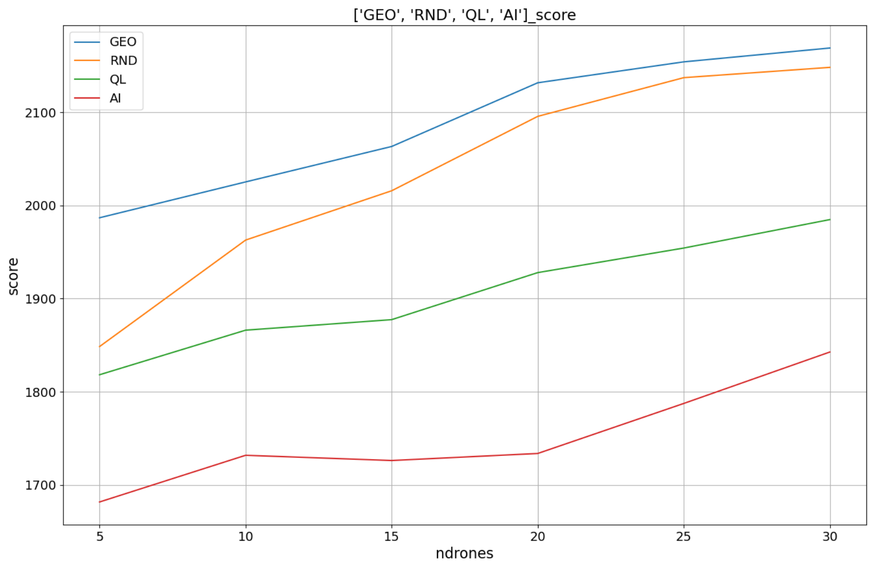

Figure 4: 50k simulation duration - score

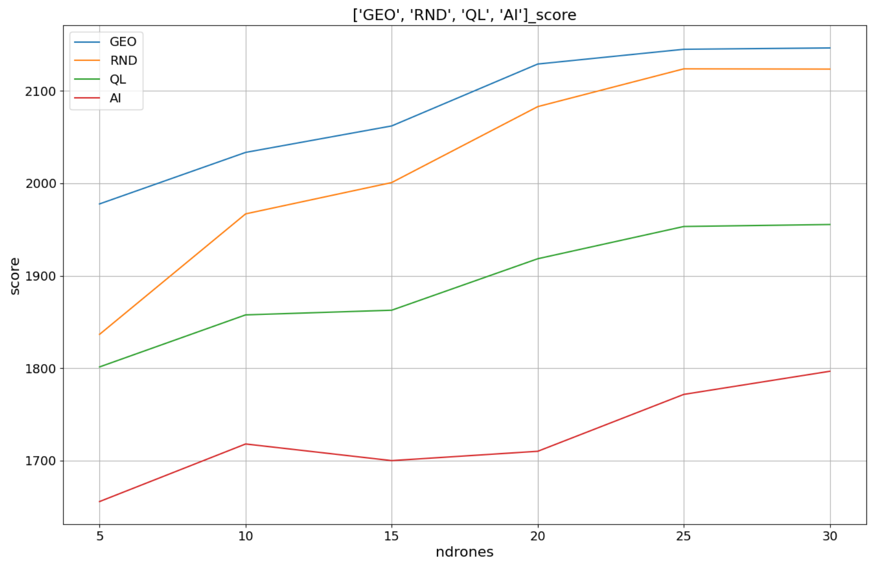

Figure 5: 100k simulation duration - score

6 Hyper-parameter tuning

We perform the tuning through grid search technique for all algorithms on 50.000 steps which is a good trade-off between 18k and 100k of sim. duration, as shown in the following figures 6. For the sake of brevity we provide plots only for ε in AI algorithm. In the plot, xAI indicate AI where $\varepsilon = 1 - x \cdot 10^{-2}$.

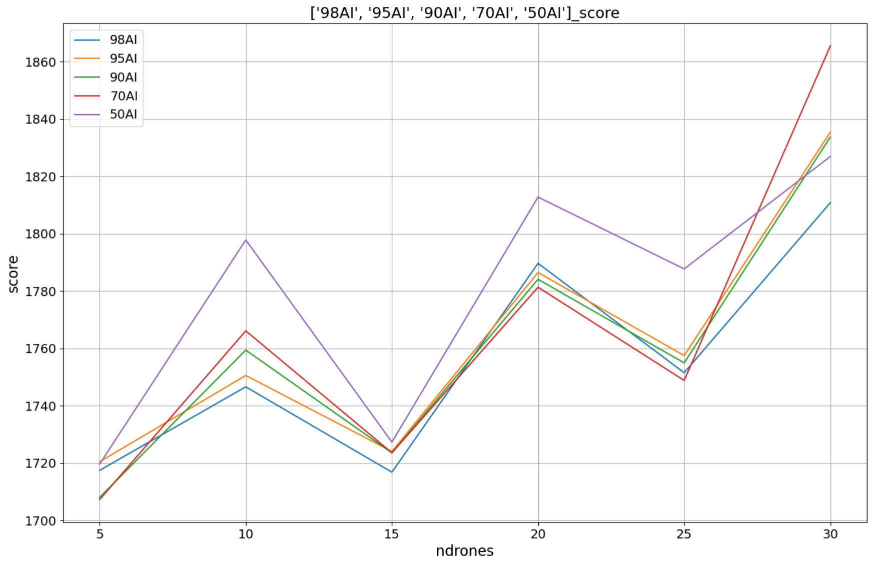

Figure 6: Hyperparameter tuning of the ε-value for the AI algorithm

We consider as best 98AI because has the behaviour more stable and gave best performance in multiple situation. The similarly happens for the QL approach.

7 Conclusion

As shown in the section 5, QL and AI outperform other algorithms with few simulation duration (18k), but we also tested in a long simulation w.r.t different scenario from 5 drones up to 30. To compare with the Professor's baseline we add the 100k simulation duration test, but we consider less and equal to 50k are enough to train our model. We deliver the AI algorithm with 50k of sim. duration but we suppose the QL approach can be also improved.

8 Contributions

The algorithm's codes have been developed in collaborative manner with scheduled meeting on Google Meet platform and using a Github repository, sharing the screen and discuss together on problems and solutions. Same approach has been used to write the report.

9 Appendix

In this section we provide the source code of the Q-learning approach:

- Q-learning: https://pastebin.com/fuPZQCFy