Homework 1 - Autonomous Networking

Andrea Bacciu
Giuseppe Masi
Rocco Pisciuneri

December 2020

Contents

1 Approaches

In this section we describe the approaches to address the MAC routing problem.

1.1 Baseline approaches

We start from a baseline approach implementing the well-know *greedy* and *ε-greedy* (obtaining different version using several update rules to estimate rewards, such as UCB and Incremental) to compare them with *Round-Robin* and *Random algorithm*. We also tried it with *optimistic initial value*.

The source codes of all algorithms implemented by us are available in the appendix section 5.

1.2 Our approaches

We implemented two strategies, both of them made use of a learning phase in which we try to understand the probability distribution or pattern of the packets over the drones.

1.2.1 Schedule approach

The schedule approach's aim to exploit the latent pattern in the drone environment. So, for each drone, in the learning phase, we allocate a fixed number of query slots (a tuned value) to learn his behaviour (considering the TTL) quering the selected drone for the entire dedicated slot to him. Once we got the learning vector for each drone, we merge them into a single one which we call *schedule*.
The schedule is a pre-computation of the action selection to use after the learning phase. To compute the schedule (as shown in Figure 1) we use the following criteria:

- **One transmission:** With $1 - \varepsilon$ probability we take the only drone that transmitted, with ε probability we pick one drone uniform at random from the set of best drones[1].

[1]The set of the best drone is the 35% (tuned) of all drones with the highest transmission rate

- **Overlap:** In the case the learned vector presents a overlap[2], we take one drone uniform at random from the set of drones in overlapping union the set of best drones.
- **No transmission:** We take one drone uniform at random from the best drones.

After the learning phase, we repeat the schedule to query drones.

Figure 1: OL: How OL build the schedule vector

1.2.2 AI

We use the learned probability distribution like a baseline to estimate reward values. As long as the task is a stationary problem, the reward values are fixed because that distribution does not change over time. During the simulation, to decide to query the last drone again, we take the *last_feedback* received from the last queried drone and we temporarily add it to the estimated reward values of that drone, as follows:

$$\mathcal{Q}_{t+1}[last\_drone] = \begin{cases} \mathcal{Q}_t[last\_drone] + int(transmission) + feedback, \\ \quad \text{if } transmission == TRUE \vee feedback > 0 \\ \mathcal{Q}_t[last\_drone]/2, \text{ otherwise} \end{cases}$$

[2]with overlap we mean overlapping the learning vector of two or more drones

This means that if a drone has lost one or more packets we increase his estimated reward value in order to query it more frequently (even if there is no transmission). In the latter case we give a penalty to the last queried drone because it didn't transmit nothing and also it has not lost packets (i.e. it is not necessary query it frequently).

Like ε-greedy approach, the action selection is done as follows: with ε probability we take one drone uniform at random; with probability $1 - \varepsilon$ we take the best drone in $\mathcal{Q}$.

2 Experiments & Benchmark

In this section we show performances of each approaches:

- RND: random;
- RR: Round-Robin;
- **INCR**: [3] Incremental Update Rule;
- **UCB**: [4] Upper Confidence Bound;
- **OL**: Schedule approach;
- **AI**

The highlighted algorithms have been developed ourselves.

In section 2.3 we test the scalability of all algorithms w.r.t. the number of drones.
To search the best ε value, we perform a grid search over a set of possible values, as shown in 2.4.

2.1 Experimental setup

All experiments in this section, unless otherwise specified, use 15.000 steps, all hyper-parameters are already tuned for each algorithm in the way shown in the section 2.4. The libraries used by the project are the python3 standard library and Numpy.

2.2 Normal instances analysis

We will show in the following plots the performance of the above algorithm on the number of drones 2, 5, 8, 10 and 15 initializing the simulation with seeds starting from 0 up to 9.

[3] ε-greedy and Optimistic Initial Value set to 9 with incremental update rule
[4] ε-greedy and Optimistic Initial Value set to 17 with UCB update rule

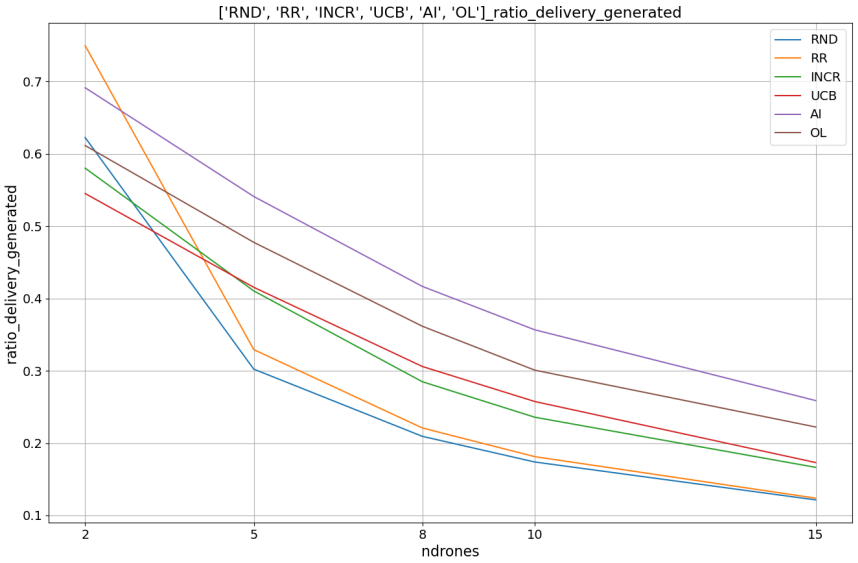


Figure 2: Gaussian probability distribution.

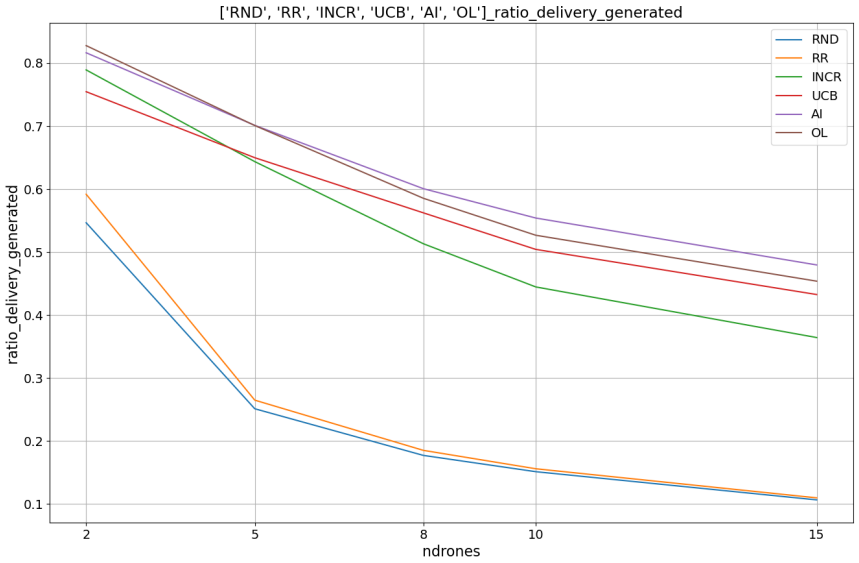


Figure 3: Fixed probability distribution.

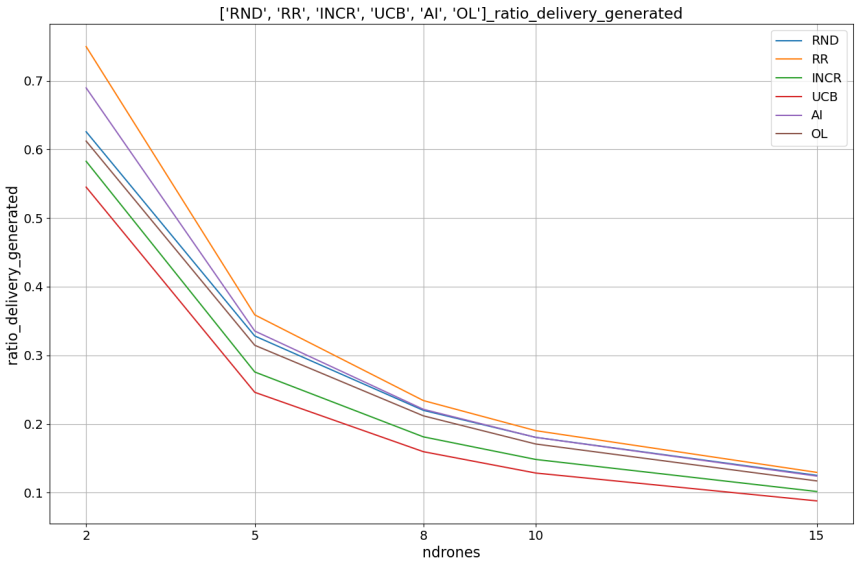


Figure 4: Uniform probability distribution.

2.3 Big instances analysis

In this section we will show the convergence also when the number of drone increases up to 100, testing it with 150.000 steps which is enough to meet the needs of all algorithm learning phase duration.

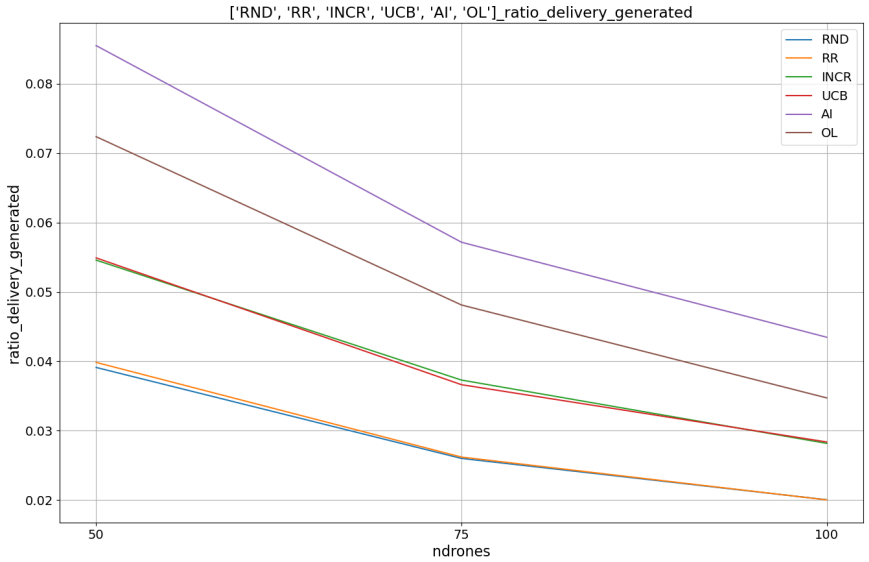


Figure 5: Gaussian probability distribution tested on 150.000 steps with up to 100 drones.

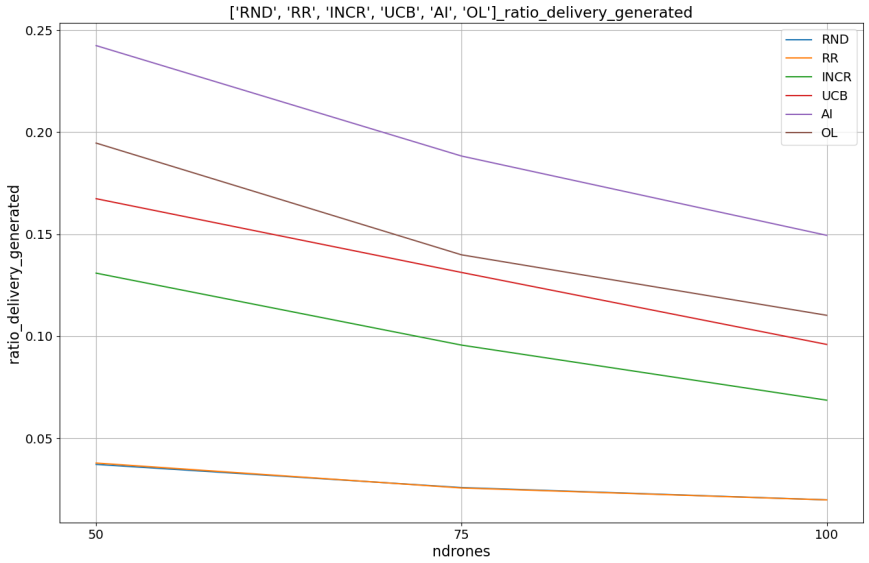
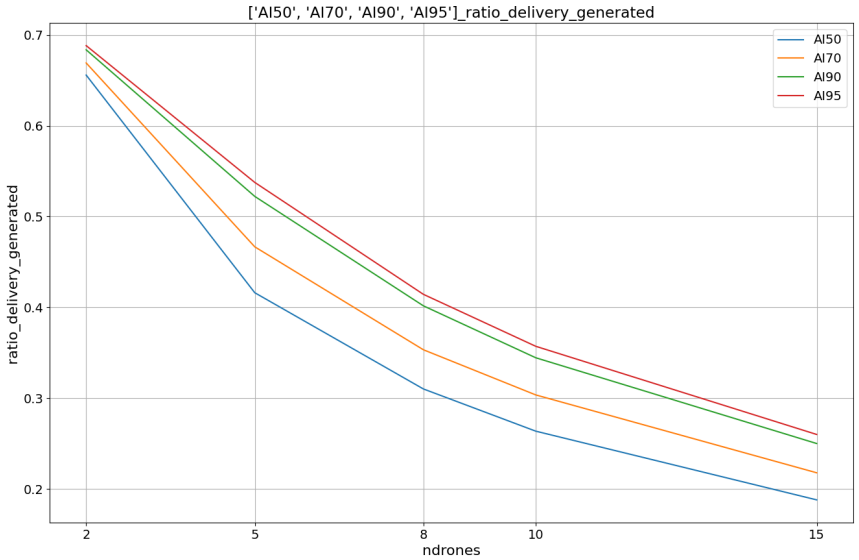


Figure 6: Fixed probability distribution tested on 150.000 steps with up to 100 drones.

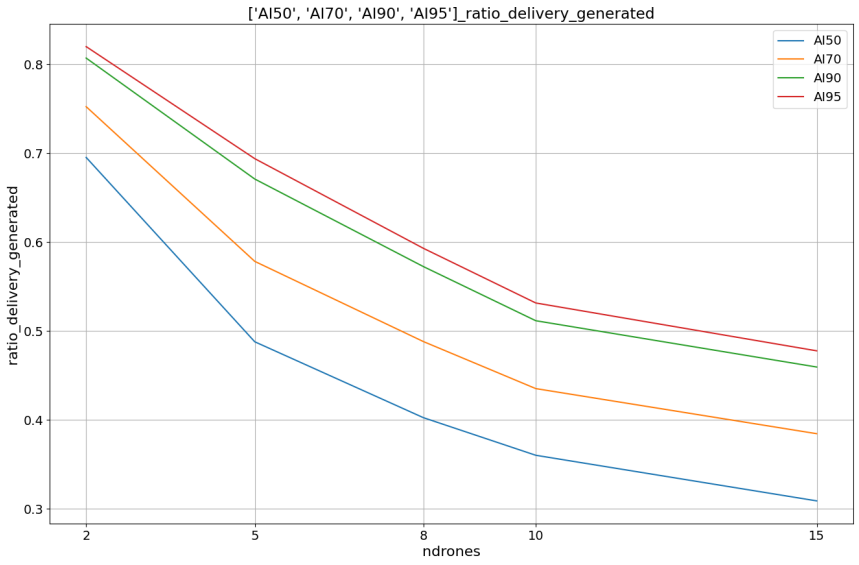
In the plots showed until now, we can clearly see that the best approach is AI.

2.4 Hyper-parameters tuning

We perform the tuning through grid search technique for all algorithms on 15.000 steps, as shown in the following figures 7. For the sake of brevity we provide plots only for ε in two distribution and for one algorithm. We have also done the same process to all other hyper-parameters such as Optimistic initial value etc.
In the plots, AIx indicate AI where $\varepsilon = 1 - x \cdot 10^{-2}$. After this analysis we chose the ε equal to 0.05, for the AI algorithm, which turned out to be the best in both gaussian and fixed probability.



(a) Gaussian probability distribution - ε tuning on AI algorithm.



(b) Fixed probability distribution - ε tuning on AI algorithm.

Figure 7: ε tuning.

3 Conclusion

All algorithms proposed avoid the starvation problem through the use of ε value. In general, AI and OL dominate other approaches on the gaussian and fixed probability distribution, as shown in section 2. Moreover, we tested a more dense environment where the number of drones increase as shown in Figure 5,6.
About the uniform distribution, as long as there is an equal distribution of packets to drones, there's no pattern to learn; despite several strategies applied there's no way to get a better performance than the random algorithm. During the project development we work on finding some recurrences in the drones communication, but we can conclude there is no one because the events are independent.
We delivered the AI algorithm but we suppose that in a real-world application OL performs better than AI, because it is able to understand and repeat the real-world pattern.

4 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.

5 Appendix

In this section we provide the source code of the other algorithms developed during the research of MAC protocols.

- INCR (ε-greedy and optimistic initial value):
  https://pastebin.com/qsGRJHTU
- UCB (ε-greedy and optimistic initial value):
  https://pastebin.com/CPk2iS7J
- OL:
  https://pastebin.com/ySBqPBhP