Homework 3 - Autonomous Networking

Andrea Bacciu
Giuseppe Masi
Rocco Pisciuneri

January 2021

Contents

1 Introduction

In this work, we address the MAC protocol problem in a distributed manner, so in contrast to the centralized version, there's no centralized entity (depot) that issues the transmission schedule to each drone in the network. In the distributed version each drone makes autonomous decisions, in which they try to synchronize themselves with other drones in the transmission channel. In the MAC problem, the time is divided into repetitive frames and each frame is slotted. Each drone can autonomously decide if communicate or not in each slot (if there's a packet to transmit). Our task aims to find a learning algorithm to let the drone to synchronize itself with other drones in the transmission media taking care of the following variables ***unused slots***, ***collisions***, ***collected ratio***. We want to maximize the utilization of the transmission media (increasing the collected ratio), so it involves in the minimization of the unused slot and collisions in order to avoid useless battery drain.

2 Related Work

From the recent works [1] et al. on WSN (Wireless Sensor Network), they propose a combination between the well-known MAC protocol Aloha and the novel approach Q-Learning resulting in the Aloha-Q algorithm with the aim of improve the power efficiency in MAC protocol. Current MAC protocols can be broadly divided into contention-based and schedule-based. The majority of the proposed schemes are contention-based and inherently distributed, but they introduce energy waste through overhearing, collisions, idle-listening, and re-transmissions [2]. Therefore, ALOHA-Q works like a contention-based scheme gradually transforming into a schedule-based scheme. ALOHA-Q aims to reach an optimal steady state where each slot in a frame is reserved to only one user. It allows users to find unique transmission slots in a fully distributed manner, resulting in a scheduled outcome. [3].

2.1 How it works

ALOHA-Q divides time into repeating frames where a certain number of slots are included in each frame for data transmission, each node can access only one slot per frame. Each slot is initiated with a Q-value to represent the willingness of this slot for reservation, which is initialized to 0 on start-up. Upon a transmission, the Q-value of corresponding slot is updated, using the Q-learning update rule given by

$$Q_{t+1}(i,s) = Q_t(i,s) + \alpha(R - Q_t(i,s))$$

where i indicates the present node, s is the slot identifier, R is the current reward. All the hyper-parameter used are reported in the table 1.

Table 1: Q-Aloha Hyperparameters

HParams	**Value**	**Symbol**
Learning rate	0.1	α
Reward	±1	R
Epsilon	0.1	ε

2.1.1 Exploration and Exploitation

The Aloha-Q made the use of ε-greedy strategy to select the action to perform. In particular, in this settings each drone can schedule in which slot will transmit a data packet with the following behavior:

- With ε probability it selects random slot;
- With 1 - ε probability it picks the slot with the highest Q-value.

Exploration is, of course, particularly crucial when the environment is non-stationary.

3 Our Approaches & Problem Modeling

The problem faced made the use of slotted time following the contention-based scheme in a pure distributed settings with no communication between drones. The time is slotted following this calculation

$$len\_frame = number\_of\_drones * \mathcal{K}$$

[1] Given a curr_step, to obtain the actual slot in a frame we follow:

$$Slot = curr\_step \ \% \ len\_frame \tag{1}$$

3.1 Communicate - Actions

In the Related Work described in the Section 2 they choose in which slot communicate a packet. Instead in the our simulator environment we can choose if transmit a packet in a slot or not. In our problem formulation all packets have the same TTL (Time To Live) so we found it beneficial to transmit first the oldest one.

In order to avoid starvation we use the ε-greedy strategy in all algorithm proposed:

- With ε probability it select randomly (50%) if transmit the packet or not;
- With 1 - ε probability it decide to transmit accordingly to the specific algorithm design.

3.2 State representation

The state representation is the current slot calculated by the formula 1.

3.3 Hyper Parameters used

All the hyper parameters used in our approaches were tuned as shown in Section 8, where we can also see the convergence of out algorithms.

4 Q-Learning approach

In a similar fashion to what was described in Section 2 we rely on a Reinforcement Learning approach combining the Q-Learning algorithm and Slotted Aloha. As we stated before (Section 3) we are in a contention-based scheme but we let the model to learn a schedule in a competitive setting to converge into a sort of schedule-based scheme. We deploy the Aloha-Q algorithm in each drone in which they own their

[1]The $\mathcal{K}$ is found after a tuning phase

not-shared Q-Table. The Q-table contains the state representation from the point of view of the agent (drone). After receiving the feedback we update the Q-value following this rule:

$$Q_{t+1}(s) = Q_t(s) + \alpha(R - Q_t(s)) \tag{2}$$

which is the same used in the original formulation of Aloha-Q [3], we only omitted the drone's index. We made the use of a function called *taken_action* able to map the packets to a given slot in order to record the link between them. As we stated in Our Approaches Section 3, the main difference between our formulation and the Aloha-Q presented in the related work Section 2 is the fact the drone made a decision at each *curr_step* if transmit or not in the base if consider the actual slot good enough to transmit. This decision relies on threshold $\mathcal{T}$ over the Q-Value for a given state, if the Q-value is greater than $\mathcal{T}$ so the drone use the slot. With the Aloha-Q formulation, this leads the model to small increments of the Q-Value for the positive feedback and a higher decrease for a penalty. This issue is also cited in the work of [3] in which they state the fact that starting from a Q-value of 1 (upper bound) with only 7 consecutive failures cause a Q-value to return to 0 at a learning rate (α) of 0.1, as the punishment has more impact on the Q-value when the Q-value is positive.

5 NAIVE approach

The NAIVE algorithm models the agent using the same notion of Aloha-Q for the Q-table and implements a re-try strategy. The name of this algorithm came from the naive intuition of the strategy trying to avoid the unbalancing between Positive and Negative reward. This algorithm is a transition from our Q-Learning approach to the BAD_GOOD algorithm which we introduce in the next Section. In this case, the Q-value is a Boolean value, so in the exploitation step if it is True the agent transmits, otherwise no. This Q-Value reflects the last opinion of the agent w.r.t the last feedback in a particular slot. In response to negative feedback for packets in a slot, we design a *re-try* strategy which consists of considering that feedback as positive according to a coin toss $\mathcal{C}$ (50% found after the tuning phase). This strategy allows the model to *re-try* another transmission on a slot even if a collision occurred previously, in order to minimize the unused slots.

6 BAD_GOOD approach

The BAD_GOOD approach is born to cope with the issue of the unbalance effect between positive and negative feedback. In this case, the Q-Table does not follow the Q-learning updating rule, instead, it records how much positive and negative feedback the drone received in a slot. As stated in Our Approaches Section 3, this algorithm also made the use of ε-greedy strategy, in particular in the exploitation step the agent performs the transmission in a slot if the quantity of positive feedback is greater than the negative ones. The main idea of this algorithm is the design of a strategy that involves a more resilient and balance managing of the feedback. So in presence of several positive feedback the history is not ruined by a few negative feedback, maybe generated by the epsilon step of another agent in the environment.

7 Benchmark & Experimental Setup

In this Section we present our experiment done in a simulation environment using the DroNETworkSimulator[2]. The table 2, 3, 4, 5 show the performance of all algorithm in all settings with 10 drones and 24k of simulation duration, using the best parameters. Calculated as the mean over the results obtained in five different simulation using distinct seeds. N.B: EMDT is the Event Mean Delivery Time. Later in this

[2] https://github.com/Andrea94c/DroNETworkSimulator

chapter, we introduce a more extensive result with plots taking care of ration delivery generated and event mean delivery time.

Table 2: Test Gaussian

Algorithm	**Collected Ratio**	**Collisions**	**Un-used slots**	**EMDT**
RND	0.1573	17769	**767**	6.82
QL	**0.7148**	2063	4197	6.53
NAIVE	0.5478	2503	7434	6.57
BAD_GOOD	0.6756	**2059**	5065	**6.39**

Table 3: Test Not-Stationary

Algorithm	**Collected Ratio**	**Collisions**	**Un-used slots**	**EMDT**
RND	0.1747	17048	**1075**	7.41
QL	**0.6393**	2485	7667	**6.5**
NAIVE	0.538	2398	7365	6.82
BAD_GOOD	0.5866	**1853**	7031	6.27

Table 4: Test Uniform

Algorithm	**Collected Ratio**	**Collisions**	**Un-used slots**	**EMDT**
RND	0.0674	22129	**252**	6.22
QL	**0.6151**	2552	6674	**5.61**
NAIVE	0.4960	3591	8496	5.69
BAD_GOOD	0.6098	**2470**	6882	**5.61**

Table 5: Test Fixed Probability

Algorithm	**Collected Ratio**	**Collisions**	**Un-used slots**	**EMDT**
RND	0.3455	12510	**3192**	7.65
QL	0.7602	**1503**	4238	**7.52**
NAIVE	0.5688	1642	8697	7.78
BAD_GOOD	**0.7664**	1540	4050	7.8

7.1 Gaussian distribution

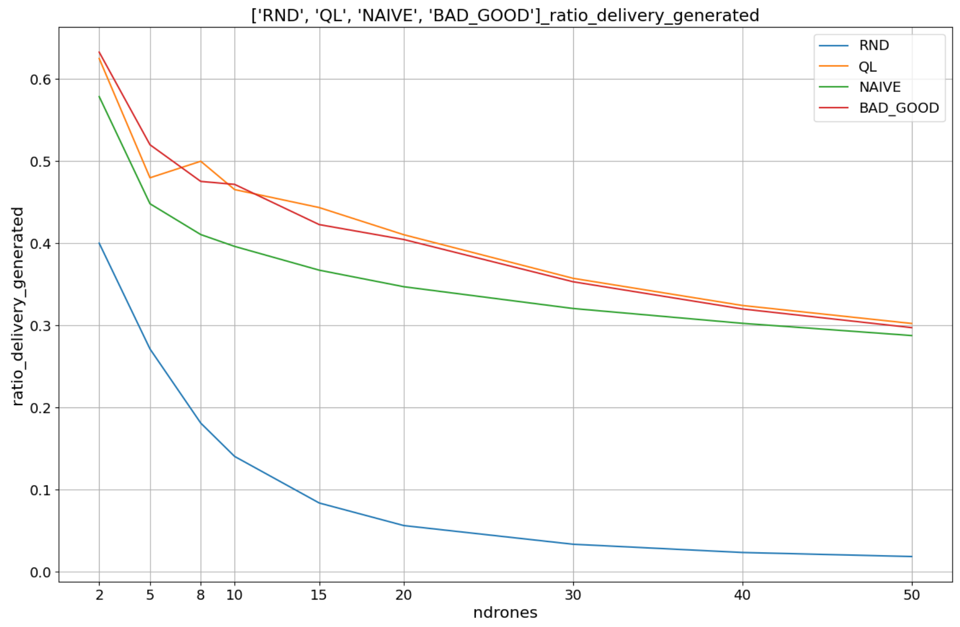

Figure 1: Gaussian Ration delivery generated

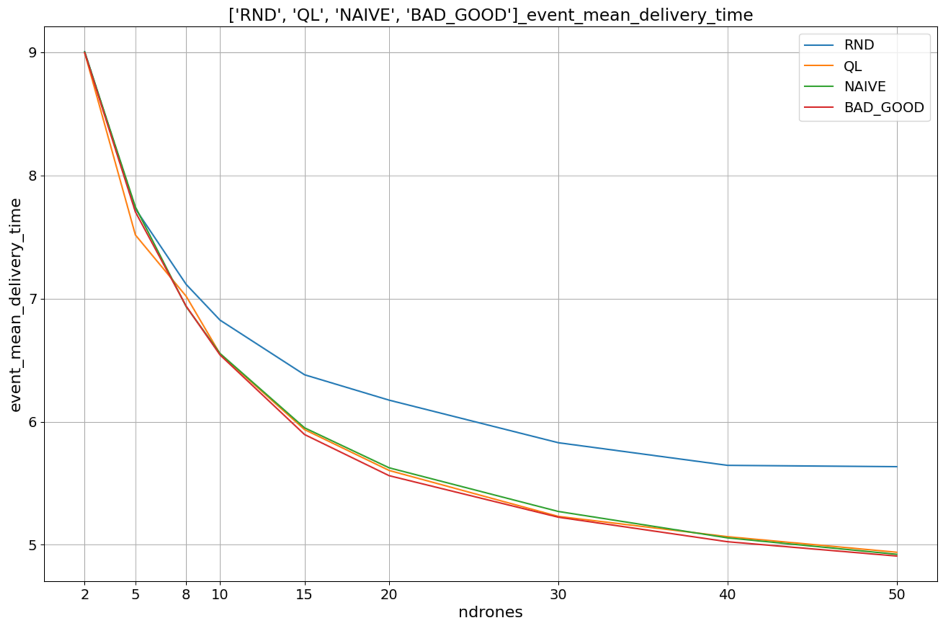

Figure 2: Gaussian Event Mean Delivery Time

7.2 Uniform distribution

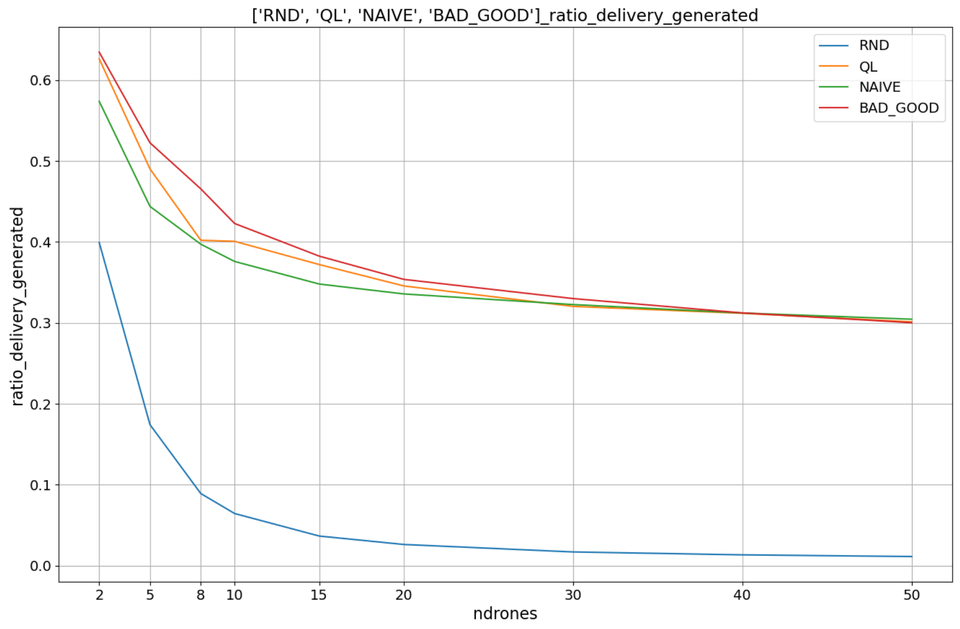

Figure 3: Uniform Ration delivery generated

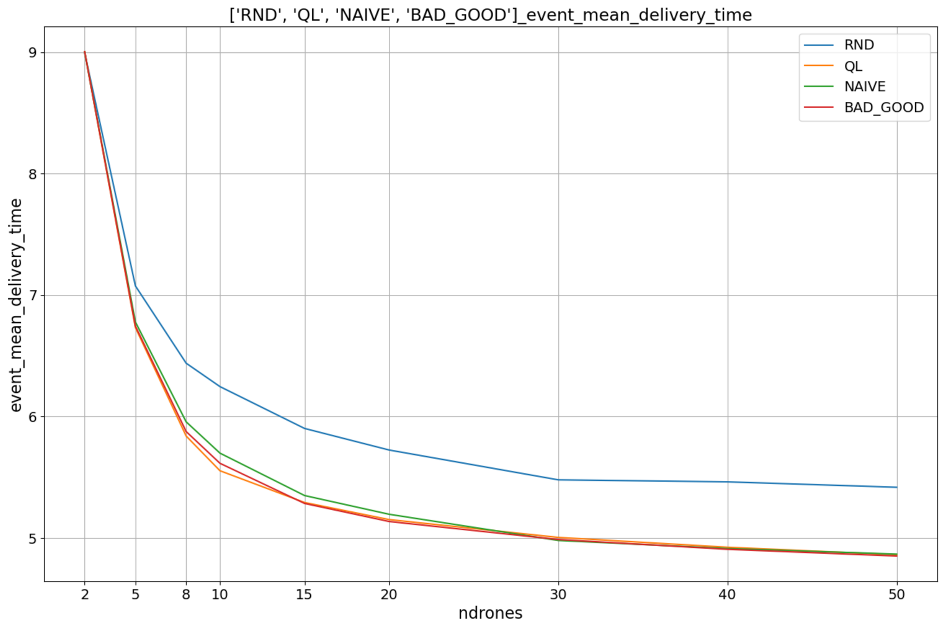

Figure 4: Uniform Event Mean Delivery Time

7.3 Fixed Probability distribution

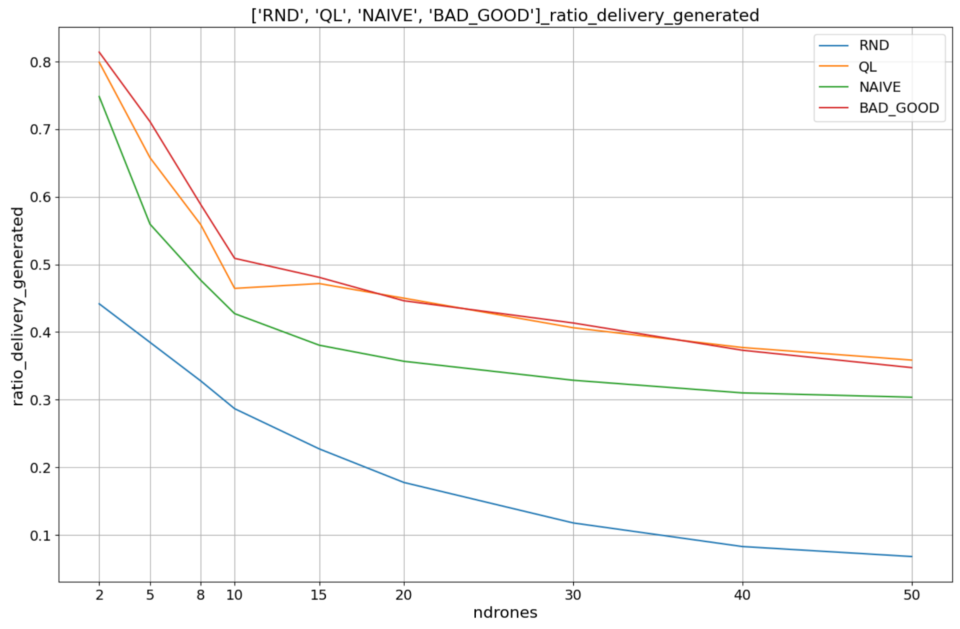

Figure 5: Fixed Probability Ration delivery generated

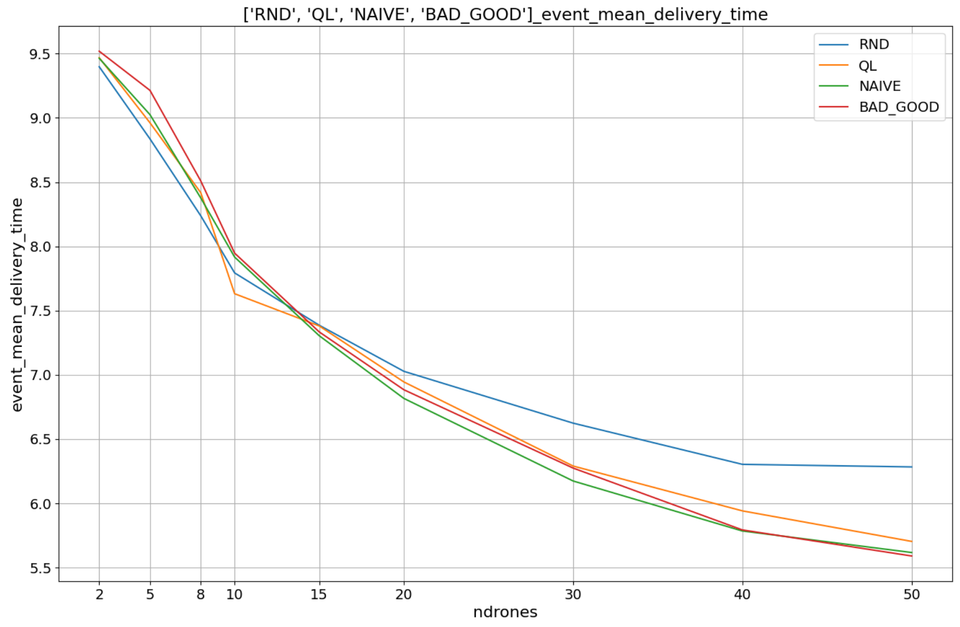

Figure 6: Fixed Probability Event Mean Delivery Time

7.4 Not Stationary distribution

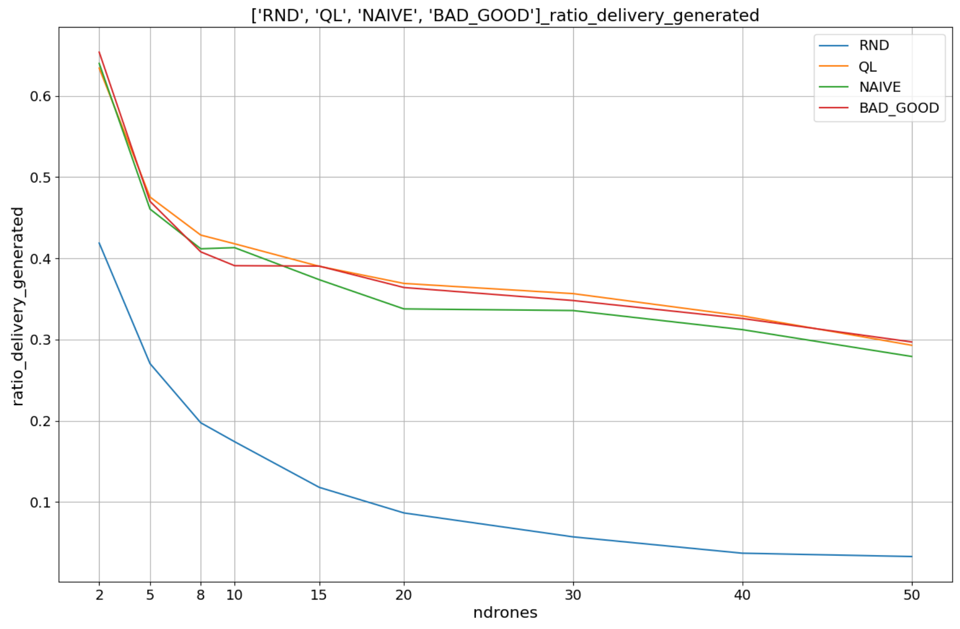

Figure 7: Not Stationary Ration delivery generated

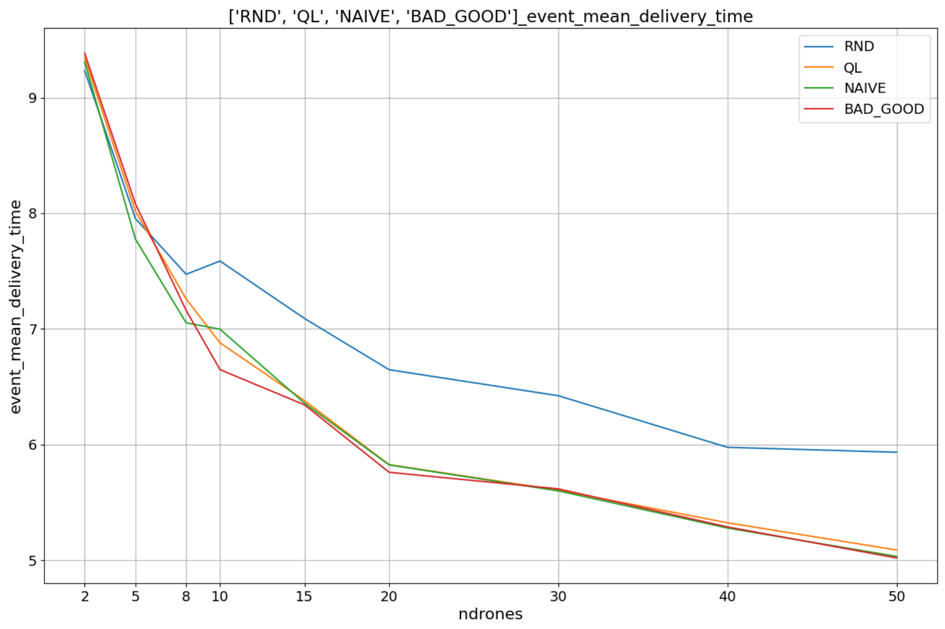

Figure 8: Not Stationary Event Mean Delivery Time

8 Hyper-parameter tuning & Convergence

In this Section, we show the Hyper-parameter tuning done and the convergence analysis on stationary and not-stationary distribution.

8.1 Tuning

We perform the tuning through grid search technique for all algorithms on 24.000 steps which are shown in the following figures 9. For the sake of brevity, we provide plots only for α in the QL algorithm. In the plot, QLx_y indicate QL where $\alpha = x, y$.

Table 6: Our Hyperparameters after the tuning phase with Random Search

HParams	**Value**	**Symbol**	**Notes**
Learning rate	0.1	α	Used only Q-Learning
Reward	± 1	R	Used only Q-Learning
Threshold	0	$\mathcal{T}$	Used only Q-Learning
Coin toss	50%	$\mathcal{C}$	Used only in NAIVE
Epsilon	0.05	ε	Used everywhere
HParam Slotted	2	$\mathcal{K}$	Used everywhere

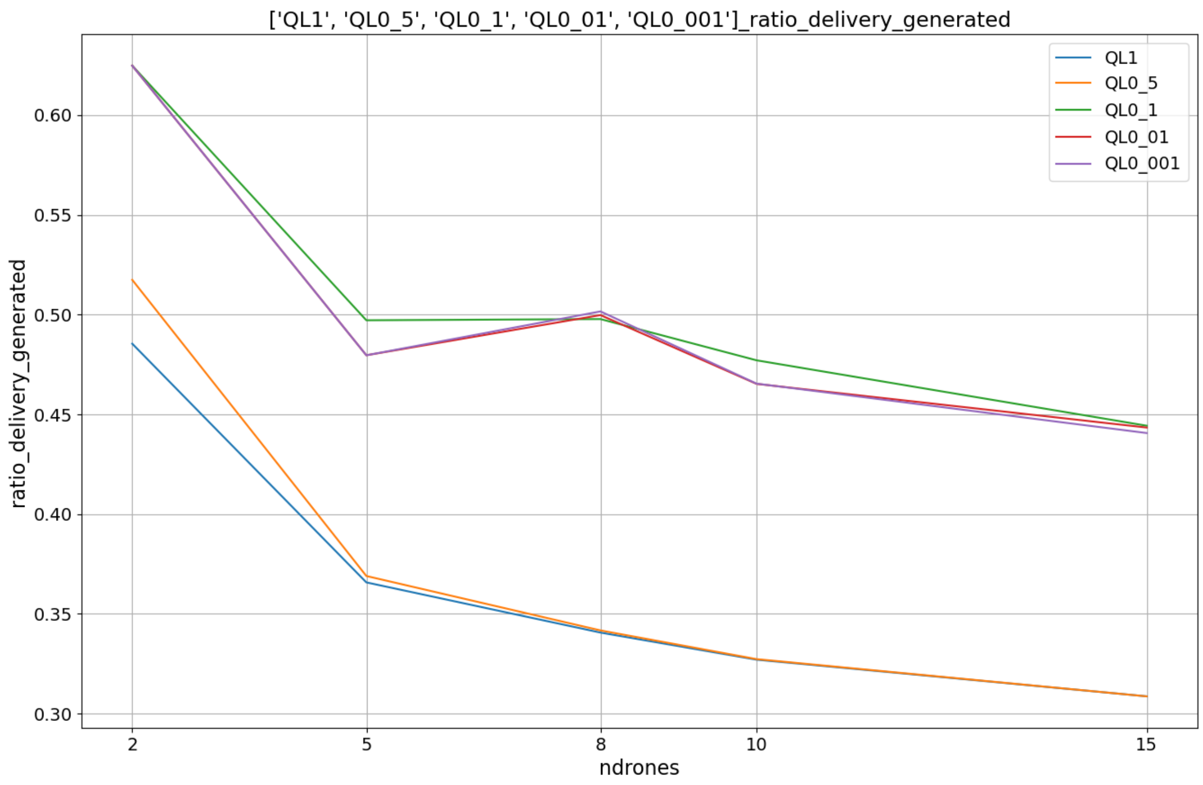

Figure 9: Hyper parameter tuning - Gaussian - Ration delivery generated, 24k sim duration

8.2 Convergence

In this subsection, we show the convergence of our algorithms, in particular, we run each algorithm with 2, 5, 8, 10, 15 drones and with 10 different seeds. For the sake of brevity, we provide plots only for the Gaussian distribution. We run the experiment with 12.000, 24.000 and 48.000 steps of simulation duration in order to define the minimum number of steps for the convergence. The plots with 12k and 48k are reported in figures 10 and 11 respectively. The plot with 24k steps of simulation duration for Gaussian is reported in figure 1.

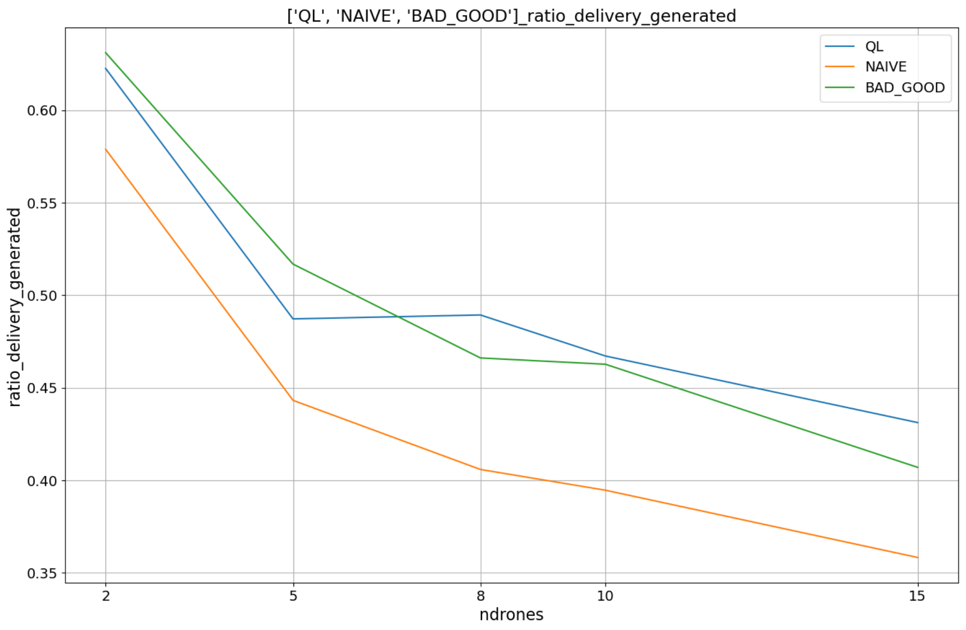

Figure 10: Convergence - Gaussian - Ration delivery generated, **12k sim duration**

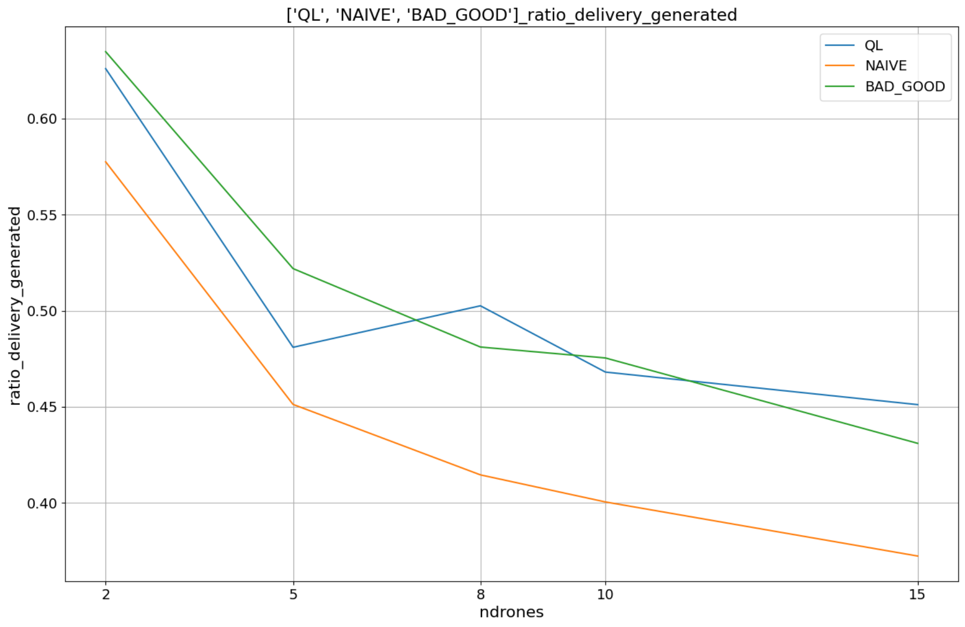

Figure 11: Convergence - Gaussian - Ration delivery generated, **48k sim duration**

At the end of the convergence analysis we can conclude that 12.000 steps of simulation duration are enough, increasing the simulation duration up to 24k and 48k we obtain a small improvement in performance but they do not justify the ratio between the improvement and the simulation duration.

9 Conclusion

As shown in the extensive tests provided in Section 7 all the MAC implementations outperform better than the Random Baseline using distinct drone squad of different sizes from 2 up to 50. We started from previous works of the Aloha-Q and we analyze the strengths and weaknesses. As highlighted by the authors is high sensibility to negative reward which lead the model to ruin their higher Q-value only with few penalties. This situation is possible to obtain by the ε-greedy strategy in which with large number of drones perform their exploration step. This issue leads to the developments of the BAD_GOOD approach that balance the effect of positive and negative feedback. However, this technique obtains close performance to the QL approach as can be seen in the plots and tables in Section 7. We consider good enough the performance of all algorithm to address the Distributed MAC problem but the best choice rely on QL and BAD_GOOD.

10 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.

11 Appendix

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

- Q-learning: https://pastebin.com/GUZEdH2W
- NAIVE: https://pastebin.com/dMZ5fCVy
- BAD_GOOD: https://pastebin.com/tuPvApwY

References

[1] Y. Chu, P. D. Mitchell, and D. Grace, "Aloha and q-learning based medium access control for wireless sensor networks," in *2012 International Symposium on Wireless Communication Systems (ISWCS)*, pp. 511–515, 2012.

[2] I. Demirkol, C. Ersoy, and F. Alagoz, "Mac protocols for wireless sensor networks: a survey," *IEEE Communications Magazine*, vol. 44, no. 4, pp. 115–121, 2006.

[3] S. Kosunalp, Y. Chu, P. D. Mitchell, D. Grace, and T. Clarke, "Use of q-learning approaches for practical medium access control in wireless sensor networks," *Engineering Applications of Artificial Intelligence*, vol. 55, pp. 146–154, 2016.