Adaptive Routing

Alexandre Rassinoux

Fall 2017

Semester Project under the supervision of Panayiotis Danassis
Artificial Intelligence Laboratory LIA - EPFL

Abstract

This report presents the development work of a modular evaluation platform implemented in Java, and capable of testing multiple various algorithms in parallel, even in multi-agent decision-making problems. The project includes the analysis of a set of stochastic, adversarial, and real-world-like scenarios with some reliable interpretations, by using measure performance tools like expected regret and cumulative reward.

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Contents

1 Introduction

In many real-world applications, decisions are made in order to maximize some expected payoff. But decisions can also bring some helpful information to improve future decisions. In reinforcement learning, the agent which interacts with an unknown environment have to face a trade-off between *exploitation* and *exploration*. Exploitation enables to use the acquired knowledge to maximize the payoff, and exploration consists in gaining new knowledge by discovering the environment.

In this project, we focus on an instance of this general problem called multi-armed bandit problem and introduced by Robbins in 1985. [1]

1.1 Motivation

In practice, multi-armed bandit problems arise frequently, for example in source routing contexts. [2]

A sequence of packets has to be routed from a source host to a destination host in an unknown network. When a packet is delivered, the host receives a feedback corresponding to the time it takes to reach the destination. The feedback follows an unknown distribution depending on the chosen path of the network.

Another example can be extrapolated from the first one : road traffic. Every morning, a student has the choice between K roads that led to school. He measures the time spent during the trip in order to improve his decision the day after. The time depends on potential traffic jams caused by other students, but also on independent conditions (e.g. quality of the road or weather conditions).

1.2 Objectives

There are many studies and papers about multi-armed bandit strategies, and algorithms have already been implemented in many languages. The theoretical results and convergence properties of these algorithms are also proved.

The aim of the project is to evaluate the behaviour of multi-armed bandit strategies in a set of various situations.

The project is threefold:

1. To design a modular evaluation platform capable of working without hard coded block.
2. To implement various decision making models and algorithms.
3. To run various simulations and evaluate performance.

1.3 Outline

The report is structured as follows. In Section 2, we explain the basics of multi-armed bandit systems and models that we use. Then, in Section 3 we describe the experimental setup and the scenarios chosen for the evaluation part. Section 4 is the presentation of the experimental results. The last couple of Sections 5 and 6 are a discussion on possible future work and a conclusion on the project.

2 The Context

In this section, we first recall multi-armed bandit problems basics with a definition and some of the tools used to measure performance of bandit policies. Then, we present the three main models of existing situations, followed by a list of bandit algorithms and reward processing that the future platform will handle.

2.1 Multi-Armed Bandit Problems

2.1.1 Definition

A multi-armed bandit, also called K-armed bandit, can be compared to a slot machine (one-armed bandit) but with more than one lever. When a player (i.e. agent) pulls a lever (i.e. arm), a reward drawn from a distribution associated to that specific lever is given. In a multi-armed bandit problem, the goal for an agent is to maximize payoff through repeated trials, by focus on the most rewarding arms. [3]

We define $X_{i,t}$ the reward given by an arm i at time t.

A strategy π is an algorithm used to choose the next arm to pull based on the history of pulls and observed rewards.

2.1.2 Measure of Performance

To analyze the behavior of an agent implementing a bandit strategy, it is necessary to compare its performance with an optimal strategy which claims to choose the best action at every steps. The *regret* of an agent for not playing always optimally is one of the most popular measure. The regret (1) and the expected regret (2) after n rounds $I_1, I_2, ..., I_n$ are defined by:

$$R_n = \max_{i=1,2,...,K} \sum_{t=1}^{n} X_{i,t} - \sum_{t=1}^{n} X_{I_{t,t}} \tag{1}$$

$$\mathbb{E}\, R_n = \mathbb{E}\left[\max_{i=1,2,...,K} \sum_{t=1}^{n} X_{i,t} - \sum_{t=1}^{n} X_{I_{t,t}} \right] \tag{2}$$

The cumulative reward helps to distinguish the best strategies in some situations. Given X_t at time t, the cumulative reward G of an agent playing a bandit strategy after n rounds is defined by :

$$G_n = \sum_{t=1}^{n} X_t \tag{3}$$

2.2 Models

2.2.1 Stochastic models

In stochastic multi-armed bandit problems, reward processes do not depend on the players actions. Successive plays of arms i yield rewards $X_{i,1}, X_{i,2}, ...$ are independent and identically distributed (*IID*) according to an unknown distribution with an expectation value. Rewards are also independent across time and for each arm, that is, given $1 \leq i < j \leq K$ and $t < t'$, the two rewards $X_{i,t}$ and $X_{j,t'}$ are *IID*.

Strategies designed for stochastic bandit problems are known to respect the following heuristic : *optimism in face of uncertainty* [4]. This principle means that, even if we do not

know yet what actions are best, the policy construct an optimistic guess as to how good the expected reward of each action is, and choose the action with the highest prediction. If the action is not good enough, then the optimistic guess will decrease this option and the strategy will choose another action.

2.2.2 Adversarial models

An adversarial bandit problem can be related as a deterministic game theory problem. [5] In this case, the *adversary* may adapt to the player past behaviour to deliver rewards according to it. The goal of an adversarial arm is to maximize total regret of agent.

Strategies designed for adversarial problems do not estimate unknown parameters of any probability distribution, because an adversarial arm do not follow any probabilistic distribution. One method consists in introducing some random information in the history of reward sequence to reduce adversary opportunity of predicting agent actions.

2.2.3 Real-world models

In most of real-world bandit problems, reward processes are neither stochastic nor adversarial and are more complex, because of the dependency on various conditions like time. In Section 1, we underline that in the traffic road model, the time (i.e. payoff) to go to school depends on decisions made by other students (e.g. two students decide to take the same road and are stuck in a traffic jam).

In the project, we assume to design a simple *real-world-like* model based on traffic road behaviour. This model enables multi-agents processing, and arms will give a reward according the number of pull requests they have at a given time t. A small reward indicate many agents pull the same arm and models a congestion in the traffic flow.

2.3 Bandit Algorithms

Bandit strategies exploit a set of algorithms that are proven to work great in stochastic or adversarial cases. In this part, we present algorithms implemented in the project. Most of algorithm versions below are based on the work of Volodymyr Kuleshov and Doina Precup (2000). [6] The first five suit well in stochastic situations while the last one is designed for adversarial problems.

2.3.1 Random

Random algorithm selects an arm by following a uniform distribution. In other words, the probability on picking an arm i in a set of K arms at next round is

$$p_i(t+1) = 1/K \tag{4}$$

2.3.2 ϵ-greedy

This algorithm is widely used for sequential decision problems. At each round $t = 1, 2, ...$, ϵ-greedy selects the arm with highest empirical mean with a probability $1 - \epsilon$, and selects a random arm with a probability ϵ. Given initial empirical reward means $\hat{\mu}_1(0), ..., \hat{\mu}_K(0)$, the probability of choosing arm i at next step is

$$p_i(t+1) = \begin{cases} 1 - \epsilon + \epsilon/k & \text{if } i = \arg\max_{j=1,...,K} \hat{\mu}_j(t) \\ \epsilon/k & \text{otherwise.} \end{cases} \tag{5}$$

2.3.3 Softmax

Softmax strategy is to pick each arm with a probability that is proportional to its empirical mean, by exploiting a Boltzmann distribution. Given initial empirical reward means $\hat{\mu}_1(0), ..., \hat{\mu}_K(0)$ and τ a temperature parameter that owns the randomness of the choice, the probability of choosing arm i at next step is

$$p_i(t+1) = \frac{e^{\hat{\mu}_i(t)/\tau}}{\sum_{j=1}^{k} e^{\hat{\mu}_j(t)/\tau}} \tag{6}$$

2.3.4 Pursuit

Pursuit algorithms exploit empirical means in order to update the probability of choosing an arm i, but each arm is treated separately. Initially, each arm has the probability $p_i(0) = 1/K$ to be chosen. Given $\beta \in (0, 1)$ the learning rate, the probabilities are re-computed as follows :

$$p_i(t+1) = \begin{cases} p_i(t) + \beta(1 - p_i(t)) & \text{if } i = \arg\max_j \hat{\mu}_j(t) \\ p_i(t) + \beta(0 - p_i(t)) & \text{otherwise} \end{cases} \tag{7}$$

2.3.5 Upper Confidence Bounds (UCB)

UCB1 is the simplest algorithm in the UCB family which is considered to implement the principle of *optimism in the face of uncertainty*. In addition to empirical means, this algorithm keeps the number of times that each arm has been played, denoted by $n_i(t)$. At the beginning, each arm is played once. Then, at round t, an arm j is picked as follows :

$$j(t) = \arg\max_{i=1...k} \left(\hat{\mu}_i + \sqrt{\frac{2 \ln t}{n_i}} \right) \tag{8}$$

2.3.6 EXP3

EXP3 stands for Exponential-weight algorithm for Exploration and Exploitation. This algorithm updates a list of weights w for each of the actions, using these weights to decide which action to take next at random. If the reward of arm i is good, w_i will increase at next step.

Initially, the weight of each arm is set to 1. Then, given $\gamma \in (0, 1)$ the factor of randomness, the probability of picking an arm i at round t is defined by :

$$p_i(t) = (1 - \gamma) \frac{w_i(t)}{\sum_{j=1}^{K} w_j(t)} + \frac{\gamma}{K} \tag{9}$$

After observing the reward $x_{i_t}(t)$ given by the picked arm i, an estimated reward can be defined by :

$$\hat{x}_{i_t}(t) = x_{i_t}(t)/p_{i_t}(t) \tag{10}$$

The list of weights is updated as follows :

$$w_j(t+1) = \begin{cases} w_{i_t}(t) e^{\gamma \hat{x}_{i_t}(t)/K} & \text{if } j = i \\ w_j(t) & \text{otherwise.} \end{cases} \tag{11}$$

2.4 Reward processing

The process of giving a reward can differ, depending the nature of environment where the arm takes place (i.e. stochastic, adversarial, or real-world). The first two arms are stochastic ways to deliver a reward, the third one refers to an adversarial process, and the last one is used to simulate a real-world problem : traffic jams.

2.4.1 Bernoulli

This arm gives a reward that is generated from a Bernoulli random variable. Given p the Bernoulli parameter, the reward of a Bernoulli arm i at time t results in

$$r_{i,t} \sim \mathcal{B}(p) \tag{12}$$

2.4.2 Gaussian

The reward processing of this arm follows a Gaussian distribution. Given (μ, σ) the Gaussian parameters, the reward of a Gaussian arm i at time t is

$$r_{i,t} \sim \mathcal{N}(\mu, \sigma^2) \tag{13}$$

2.4.3 Adversarial

In this project, we model a kind of adversarial arm based on the history of agents. The reward is given according its new information, with the objective of maximizing the agent regret. The following pseudo-code explains reward processing for an arm i:

Data: agent history
Result: give computed reward
if *50 first rounds* **then**
  reward = 1 ;
else
  compute frequency of occurrences of arm i in agent history;
  if *frequency >= 90% OR frequency <= 10%* **then**
    reward = 1 - frequency
  else
    reward = frequency
  end
end

Algorithm 1: Adversarial algorithm

2.4.4 Traffic-like

To model a simple real-world model that imitate the behaviour of traffic flows, we design an arm which rewards depends on the number of requests they get at time t : the more an arm get requests, the smaller is the reward. Note that the same payoff is given to agents that requested it. Given $n_{r,t}$ the number of requests at round t, the reward of a traffic-like arm i results in

$$r_{i,t} = \frac{1}{n_{r_t}} \tag{14}$$

3 Experimental Setup

We present here the development work carried out for this project. We begin by describing in details the evaluation platform implemented. Then, we explain the different scenarios chosen for testing purposes.

3.1 The Platform

The evaluation platform is implemented in Java from scratch. It is an open-source project available on Github.

3.1.1 Architecture

The platform has a client-server infrastructure designed for multi-agent simulations. The logical architecture is illustrated in Figure 1.

Three main nodes are implemented :

- *Simulation node* : this node runs like a server and waits for connections of other nodes (i.e. agents and arms) before starting the simulation. It is in charge of measure performance, and enables the communication between agents and arms.
- *Agent node* : this node is a client and connects on simulation node when it is started. When the simulation starts, agent will run a bandit algorithm according its initial configuration.
- *Arm node* : this node is a client and connects on simulation node when it is started. When the simulation starts, arm will give a reward according its initial configuration.

3.1.2 Simulation process

Initially, the simulation node is started. Then, it will wait for connections of both agents and arms. The predefined configuration for nodes is depicted in Table 1.

The simulation node can start when all agents and arms are connected and correctly registered (i.e. the simulation node gives a unique identifier to clients that means they are connected without error). Note that every arm sends to the simulation node their expected values during the registration process.

A set of messages are sent to communicate between nodes. The following sequence shows the flow of a simulation step for one round:

1. The agent node run its bandit algorithm and send its pull request containing the identifier of the selected arm to the simulation node.
2. The simulation node awaits for all agent pull requests. Then, in an stochastic situation, it simply sends the agent pull requests to the corresponding arms. In a adversarial case, it adds arms history of the requested agent. If the simulation is traffic-like, an arm will receive a pull request with the number of agents that selected it.
3. Each arm node runs its own reward process, and send the numerical value in a response message (i.e. pull response) to the simulation node, with the agent identifier as destination address.
4. The simulation node awaits for all arm pull responses. Then, it performs measure performance by saving all reward values, and computing regret for all agents. At the end, simulation node sends pull responses to the corresponding agents which take in account the new information for the next step.

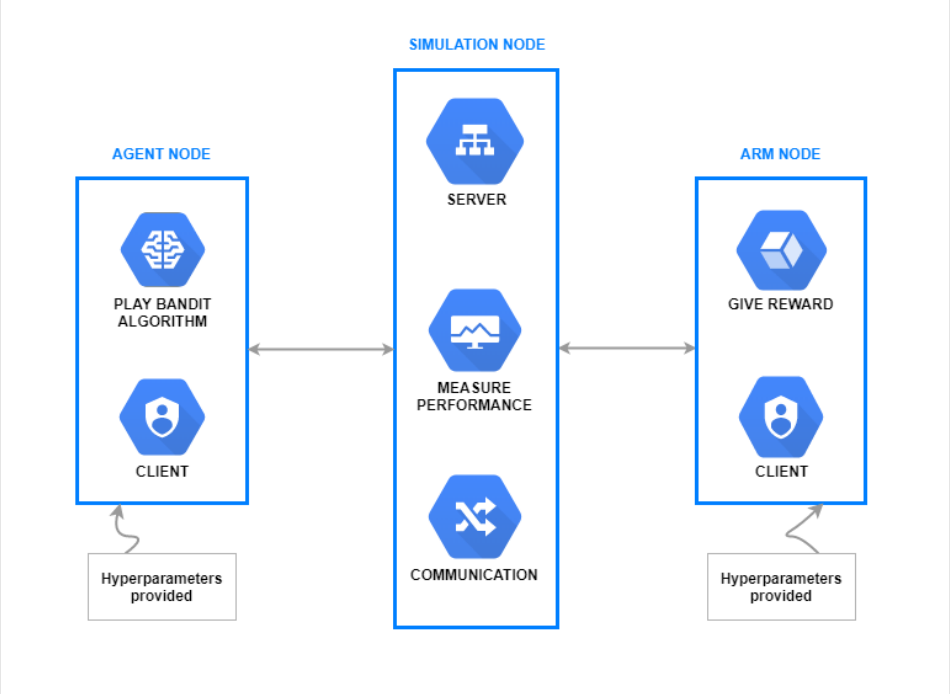

Figure 1: Logical platform architecture

3.2 Experiments

3.2.1 Overall

Even if the platform enables every possible configurations, we decide to run, for each experiment, 10 independent simulations sequentially. A simulation is composed of 500 rounds, as it is enough to all bandit algorithms to reach regret plateau. At each step, an arm reward is always chosen on the interval $[0, 1]$. The simulation node reports performance criteria at the end of the experiment in a CSV file : regret and reward per turn.

3.2.2 Scenarios

After testing many scenarios, we decide to only keep those which behaviour seems interesting to study and comment. The following paragraphs explain briefly the various experiments. For more information on the test plans, refer to Table 2 and 3 in order to get all configuration.

Impact of variance of arms As a first approach, we test the impact of the distance between the expectation values of two Bernoulli arms, in order to evaluate if algorithms still achieve to find the best arm.

Impact of number of arms This second experiment will show that the number of arms can affect the performance of a stochastic algorithm, by testing with $K = 2, 5, 10$ arms.

Comparison of stochastic bandits This experiment will compare the performance of four stochastic algorithms : ϵ-greedy, Softmax, Pursuit and UCB1, and Random, by

testing with $K = 2, 5, 10$ arms. Note that bandit hyper-parameters were already tuned for maximum performance in this case.

Performance of EXP3 in an adversarial case Here, we compare the performance of EXP3 over stochastic algorithms in an adversarial situation, with two arms.

Impact of multi-agents in traffic-like situation This last experiment evaluates the impact of multi-agents on the performance, by using arms that reproduce a simplified road traffic model.

Table 1: Required configuration for nodes

Simulation node	
Parameter	Description
requiredAgents	Agents count required in the simulation
requiredArms	Arms count required in the simulation
maxSteps	Number of simulation steps
simulationCount	Number of simulations to run

Agent node	
Parameter	Description
banditAlgorithm	The bandit algorithm the agent will use. Values accepted: random, epsilon-greedy, softmax, pursuit, UCB1, EXP3
banditHyperParameters	The hyperparameters of the bandit algorithm. For example, if ϵ-greedy is chosen, the ϵ value is required.

Arm node	
Parameter	Description
armType	The type of the arm Values accepted: Bernoulli, uniform, gaussian, adversarial, traffic-like
armHyperParameters	The hyperparameters of the arm. For example, if Bernoulli is chosen, the p value is required.

Table 2: Full configuration plans for experiments 1,2,3

Experiment 1 : Impact of variance of arms				
Sub-experiment ID	Simulation count	Rounds per simulation	Agents	Arms
1A	10	500	- ϵ-greedy, ϵ=0.01 - ϵ-greedy, ϵ=0.1 - ϵ-greedy, ϵ=0.3 - ϵ-greedy, ϵ=0.5 - ϵ-greedy, ϵ=0.7	- Bernoulli, p=0.2 - Bernoulli, p=0.8
1B	10	500	- ϵ-greedy, ϵ=0.01 - ϵ-greedy, ϵ=0.1 - ϵ-greedy, ϵ=0.3 - ϵ-greedy, ϵ=0.5 - ϵ-greedy, ϵ=0.7	- Bernoulli, p=0.4 - Bernoulli, p=0.6

Experiment 2 : Impact of number of arms				
Sub-experiment ID	Simulation count	Rounds per simulation	Agents	Arms
2A	10	500	- ϵ-greedy, ϵ=0.01 - ϵ-greedy, ϵ=0.1 - ϵ-greedy, ϵ=0.3 - ϵ-greedy, ϵ=0.5 - ϵ-greedy, ϵ=0.7	- Bernoulli, p=0.4 - Bernoulli, p=0.6
2B	10	500	- ϵ-greedy, ϵ=0.01 - ϵ-greedy, ϵ=0.1 - ϵ-greedy, ϵ=0.3 - ϵ-greedy, ϵ=0.5 - ϵ-greedy, ϵ=0.7	- Bernoulli, p=0 - Bernoulli, p=0.2 - Bernoulli, p=0.4 - Bernoulli, p=0.6 - Bernoulli, p=0.8
2C	10	500	- ϵ-greedy, ϵ=0.01 - ϵ-greedy, ϵ=0.1 - ϵ-greedy, ϵ=0.3 - ϵ-greedy, ϵ=0.5 - ϵ-greedy, ϵ=0.7	- Bernoulli, p=0 - Bernoulli, p=0.2 - Bernoulli, p=0.4 - Bernoulli, p=0.6 - Bernoulli, p=0.8 - Bernoulli, p=0 - Bernoulli, p=0.2 - Bernoulli, p=0.4 - Bernoulli, p=0.6 - Bernoulli, p=0.8

Experiment 3: Comparison of stochastic bandits				
Sub-experiment ID	Simulation count	Rounds per simulation	Agents	Arms
3A	10	500	- random - ϵ-greedy, ϵ=0.1 - softmax, τ=0.1 - pursuit, β=0.1 - UCB1	- Gaussian, μ=0.5, σ=0.1 - Gaussian, μ=0.8, σ=0.1
3B	10	500	- random - ϵ-greedy, ϵ=0.1 - softmax, τ=0.1 - pursuit, β=0.1 - UCB1	- Gaussian, μ=0, σ=0.1 - Gaussian, μ=0.2, σ=0.1 - Gaussian, μ=0.4, σ=0.1 - Gaussian, μ=0.6, σ=0.1 - Gaussian, μ=0.8, σ=0.1
3C	10	500	- random - ϵ-greedy, ϵ=0.1 - softmax, τ=0.1 - pursuit, β=0.1 - UCB1	- Gaussian, μ=0, σ=0.1 - Gaussian, μ=0.2, σ=0.1 - Gaussian, μ=0.4, σ=0.1 - Gaussian, μ=0.6, σ=0.1 - Gaussian, μ=0.8, σ=0.1 - Gaussian, μ=0, σ=0.1 - Gaussian, μ=0.2, σ=0.1 - Gaussian, μ=0.4, σ=0.1 - Gaussian, μ=0.6, σ=0.1 - Gaussian, μ=0.8, σ=0.1

Table 3: Full configuration plans for experiments 4,5

Experiment 4 : Performance of EXP3 in an adversarial case				
Sub-experiment ID	Simulation count	Rounds per simulation	Agents	Arms
4A	10	500	- random - ϵ-greedy, ϵ=0.1 - UCB1 - EXP3, γ=0.3	- Adversarial - Adversarial

Experiment 5 : Impact of multi-agents in traffic-like situation				
Sub-experiment ID	Simulation count	Rounds per simulation	Agents	Arms
5A	10	500	- random - ϵ-greedy, ϵ=0.1 - softmax, τ=0.1 - pursuit, β=0.1 - UCB1 - EXP3, γ=0.3	- Traffic - Traffic - Traffic
5B	10	500	- random - random - random - random - ϵ-greedy, ϵ=0.1 - EXP3, γ=0.3	- Traffic - Traffic - Traffic
5C	10	500	- ϵ-greedy, ϵ=0.1 - ϵ-greedy, ϵ=0.1 - ϵ-greedy, ϵ=0.1 - ϵ-greedy, ϵ=0.1 - ϵ-greedy, ϵ=0.1	- Traffic - Traffic
5D	10	500	- EXP3, γ=0.3 - EXP3, γ=0.3 - EXP3, γ=0.3 - EXP3, γ=0.3 - EXP3, γ=0.3	- Traffic - Traffic

4 Experimental Results

In this Section, we find all plots and comments on each of experimental result given by the platform. Note that the following graphs have a unique identifier in the label corresponding to a sub-experiment inside test plans depicted in Tables 2 and 3, if some other information in required.

4.1 Impact of variance of arms

We firstly observe that the ϵ parameter of bandit algorithm affect the regret curve : the smaller is ϵ, the smaller is the regret. Picking $\epsilon > 0.5$ results in more exploration than exploitation, that can explain a lack of efficiency. Also, we note that for lower values like $\epsilon = 0.01$, it implies a high standard deviation through simulations.

When the difference between the expected values of arms is quite large (Figure 2 and 3), all ϵ-greedy algorithms achieve to converge on the highest one. However, when this difference is small, as shown in Figures 4 and 5, all bandits have more difficulties to distinguish the best action to pick, that results in quite high standard deviations through simulations.

4.2 Impact of arms count

As a first approach, with $K = 2$ arms, all bandits converge very quickly to their best choice. Regret value is low since the beginning of the simulation (Figures 4 and 5).

When $K = 5$ arms, the regret increase for all algorithms, especially for those which have a high ϵ value, as we see in Figures 6 and 7. Tuning hyper-parameters can be important in case we have $K > 2$ arms.

When we have $K = 10$ arms, we do not see much differences with the previous case in term of efficiency. However, having a small value of ϵ with a high number of arms can be a problem and affect the performance of the bandit (it converges to the wrong arm in some simulations). Figures 8 and 9 reflect the standard deviation problem for too small values.

4.3 Comparison of stochastic bandits

In this experiment, algorithms were tuned for maximum performance according previous tests before starting the simulation in order to compare them properly. We also add a agent that plays the random algorithm to see the efficiency of bandit algorithms.

In a simple case where $K = 2$ Gaussian arms, pursuit algorithm outperforms all of its competitors, surely because it exploits very quickly the best arm and do not explore much during the simulation. ϵ-greedy and Softmax bandits come second with good results. (Figures 10 and 11).

When we add 3 more arms ($K = 5$) in the experiment, the ranking remains unchanged. Results are illustrated in Figures 12 and 13. It is to note that pursuit algorithm do not achieve to pick the optimal arm on each simulations, explained by a higher standard deviation.

In the situation with $K = 10$, we observe that the performance of UCB1 algorithm degrades rapidly as the number of arms increases. Pursuit algorithm remains the most powerful bandit, with standard deviation problem like in the previous case (Figures 14 and 15).

We can conclude on this experiment that ϵ-greedy and Softmax are quite stable on each situation with good results and low standard deviation. Pursuit sometimes fails but still remains the best one to minimize its regret. UCB1 lags behind its competitors, even if the results are stable through simulations.

4.4 Performance of EXP3 in an adversarial case

In this experiment, we use $K = 2$ adversarial arms. As shown in Figures 16 and 17, the EXP3 bandit outperforms stochastic and random algorithms from $t = 250$ until the end of the simulation. An advanced algorithm like UCB1 have almost the same behaviour and results as the random bandit, implying that in such case it is not efficient. ϵ-greedy, a simpler heuristic, offers better results than UCB1, especially when $t \in [50, 250]$.

EXP3 is proved to be the best alternative when we are faced such an adversarial situation.

4.5 Impact of multi-agents in traffic-like situation

To begin with our multi-agent simulations, we first evaluate the behaviour of the set of algorithm we implemented (both stochastic and adversarial). With $K = 3$ traffic-like arms, ϵ-greedy and pursuit algorithms seem to be the most efficient. We can also observe that UCB1 bandit, from round $t = 350$, has its regret curve decreasing to approach the lowest ones (Figures 18 and 19).

Then, we tend to evaluate the difference between one stochastic and one adversarial bandit behaviour, by adding random agents which are supposed to disturb the decision-making process of the two main algorithms. We already have the same number of traffic arms ($K = 3$), and we observe, as illustred in Figures 20 and 21, that both ϵ-greedy and EXP3 achieve to get lower regret than random agents, but the difference is not significantly important. That is to say that, in a multi-agents system, the nature of other agents appears to affect strongly the performance of a given bandit.

Having this information, the next step of the experiment is to know more about the behaviour when we have an even number of arms (i.e., $K = 2$), with an odd number of agents (i.e., 5) of both stochastic and adversarial nature.

When all agents are ϵ-greedy (Figures 22 and 23), we deduce that three of them picked one arm, and the two others select the other arm, that implies the first arm to deliver less reward during time. Also, we notice a strong difference in cumulative rewards between the two groups. In this case, the number of agents in this multi-agents system affect the performance of algorithms. However, when all agents are EXP3 (Figures 24 and 25), there is not two distinctive groups like in the last situation, and cumulative rewards are close. Using adversarial bandit enables each agent to have a similar reward, not like in previous case where only a small set of agents fare better.

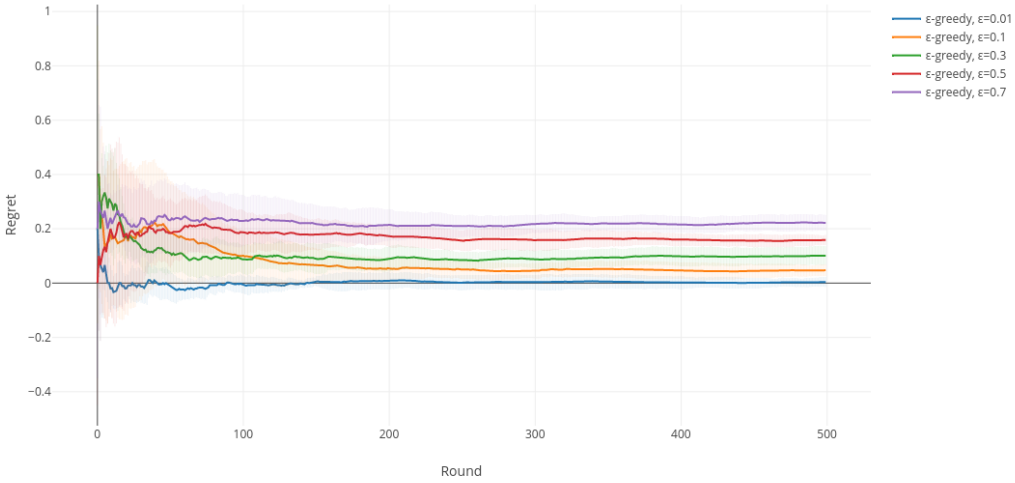

Figure 2: Regret per round - 1A experiment

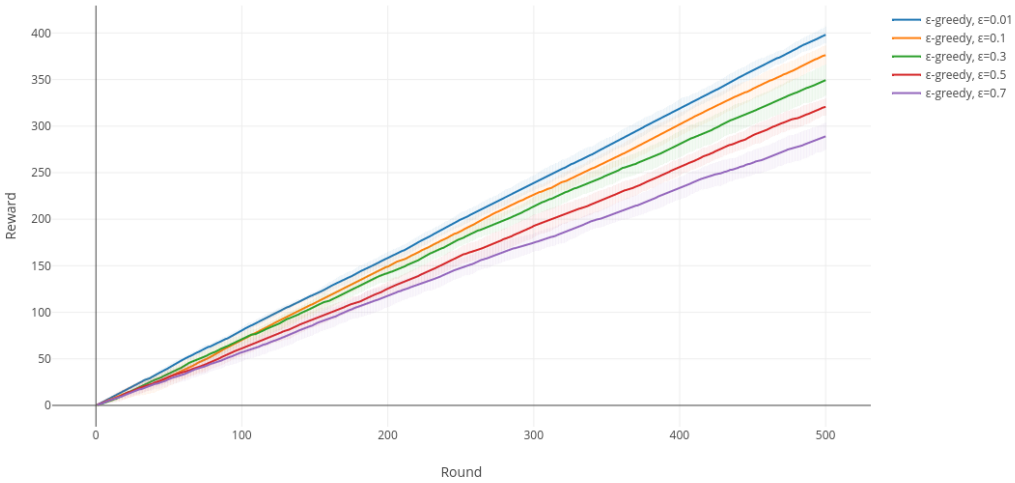

Figure 3: Cumulative rewards per round - 1A experiment

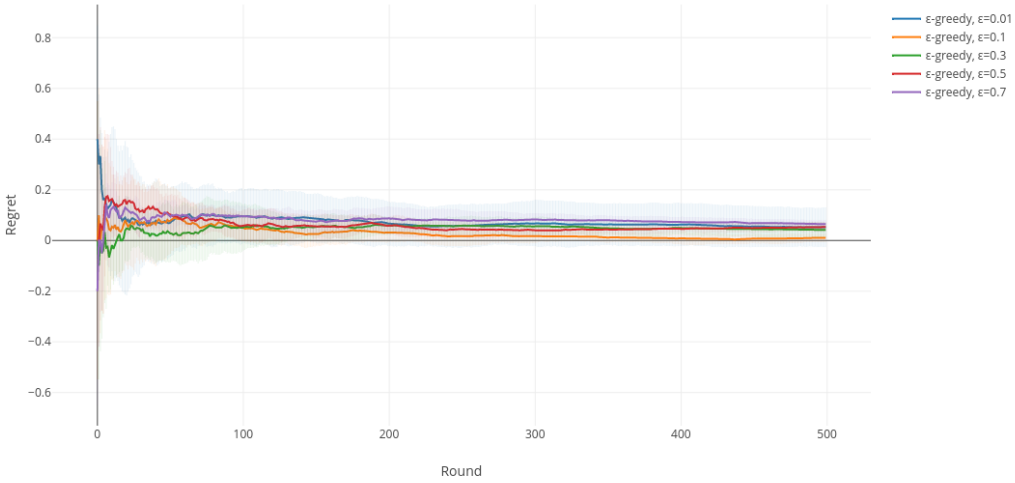

Figure 4: Regret per round - 1B and 2A experiments

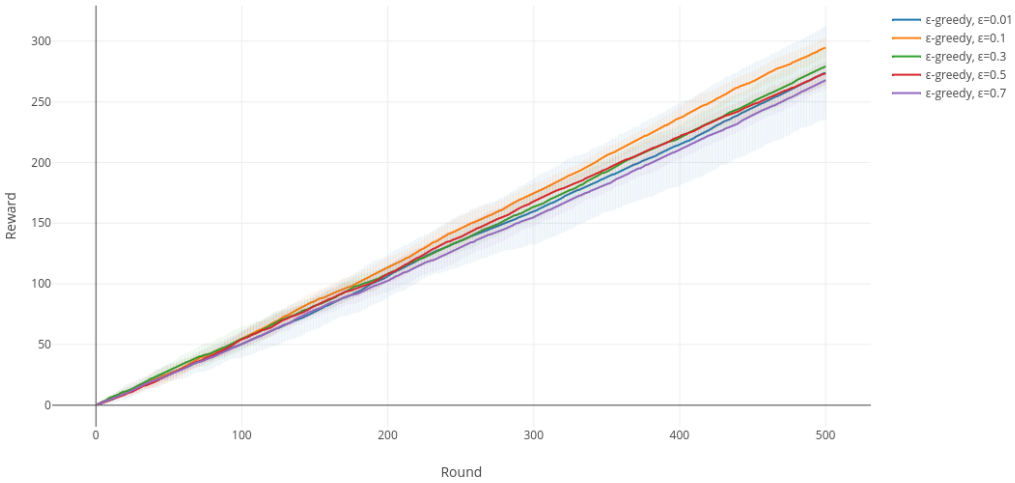

Figure 5: Cumulative rewards per round - 1B and 2A experiments

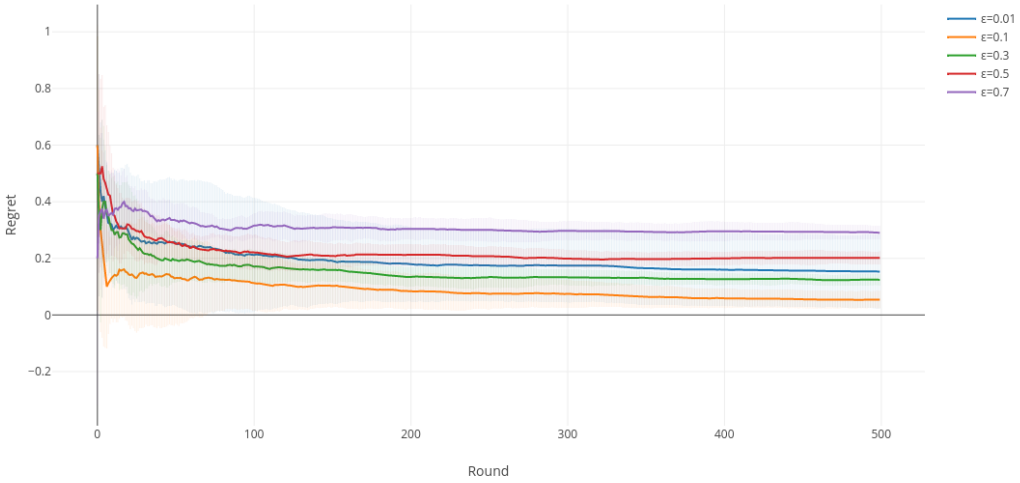

Figure 6: Regret per round - 2B experiment

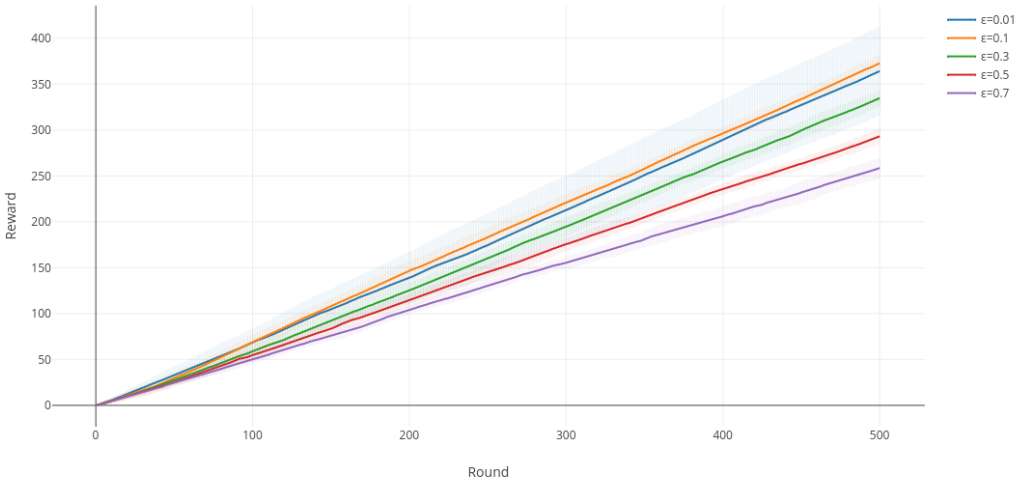

Figure 7: Cumulative rewards per round - 2B experiment

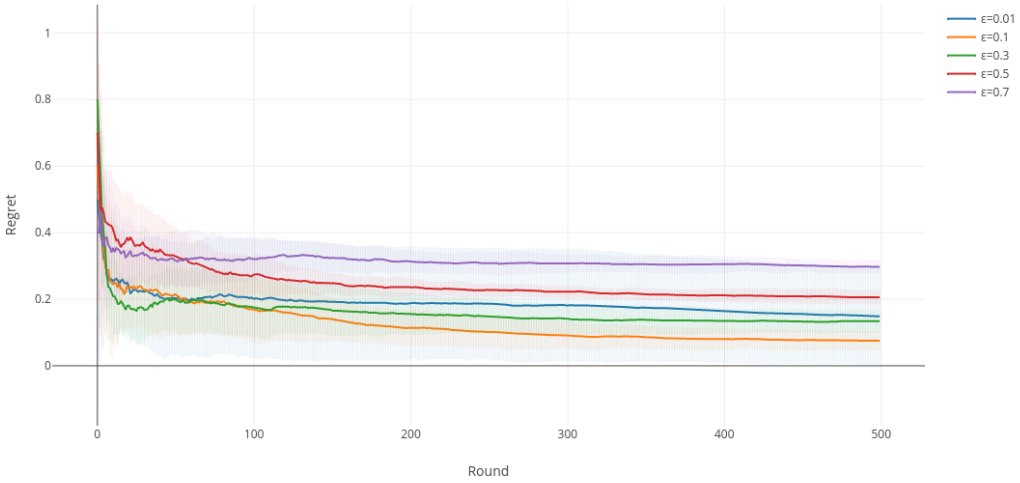

Figure 8: Regret per round - 2C experiment

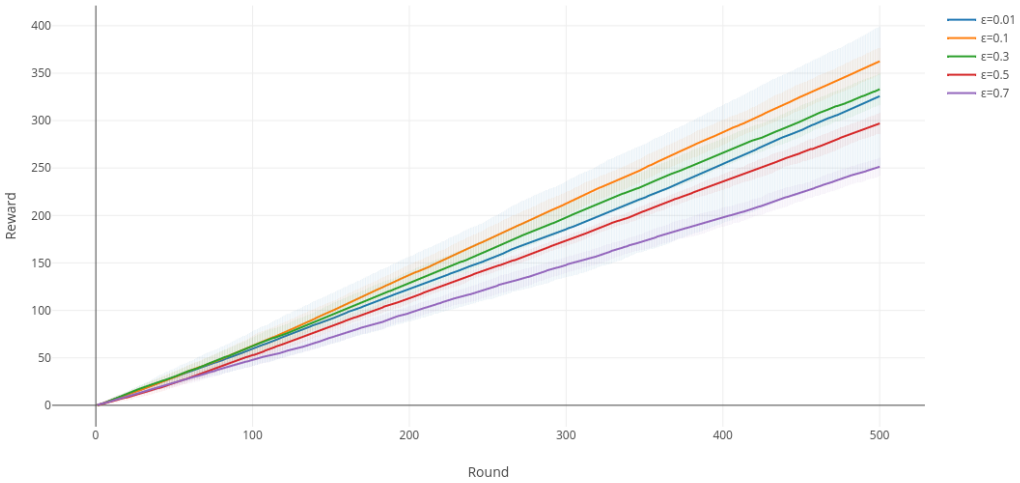

Figure 9: Cumulative rewards per round - 2C experiment

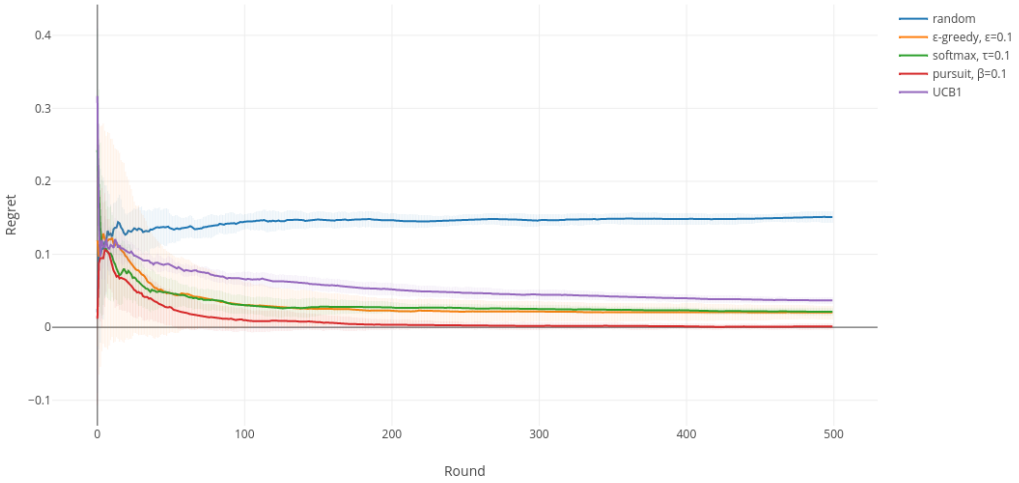

Figure 10: Regret per round - 3A experiment

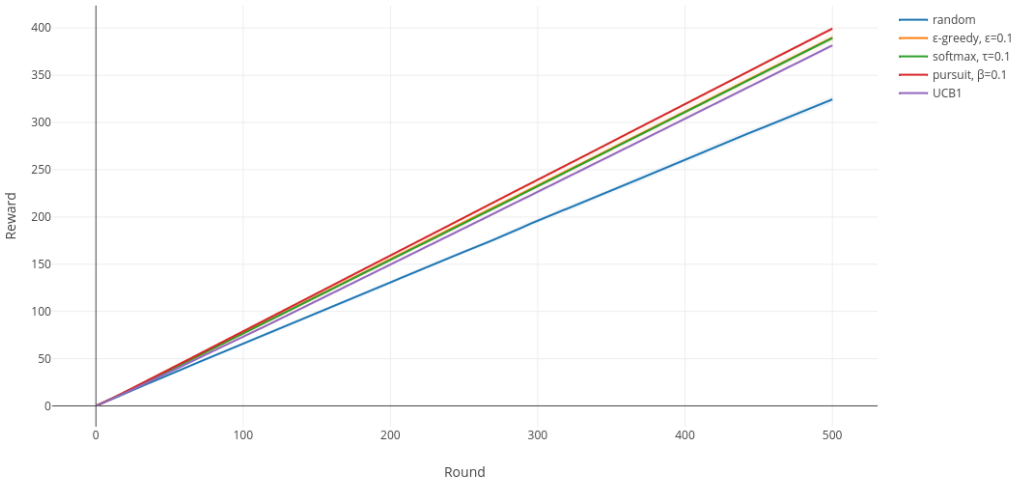

Figure 11: Cumulative rewards per round - 3A experiment

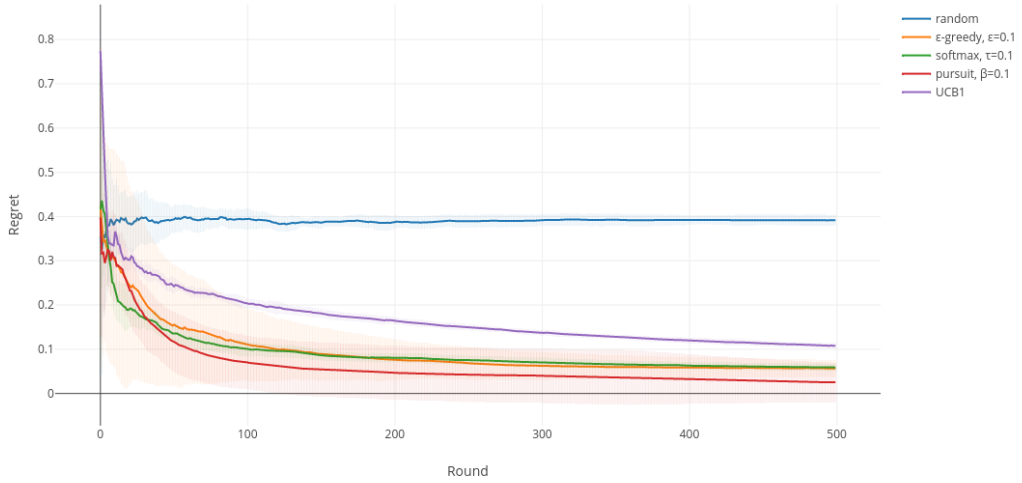

Figure 12: Regret per round - 3B experiment

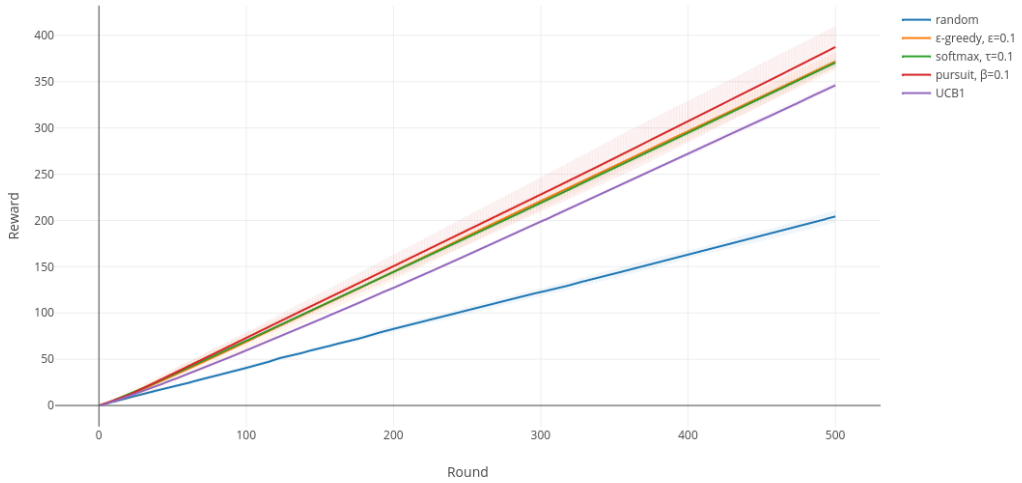

Figure 13: Cumulative rewards per round - 3B experiment

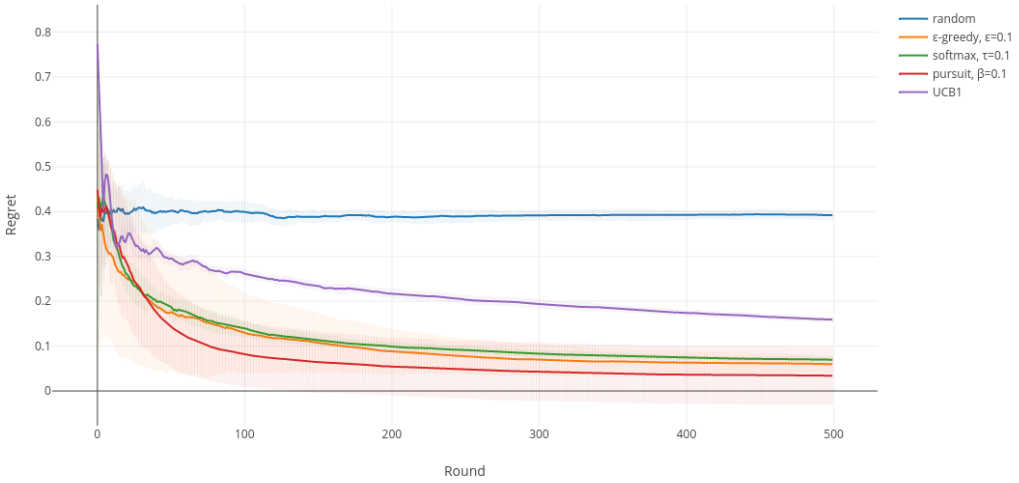

Figure 14: Regret per round - 3C experiment

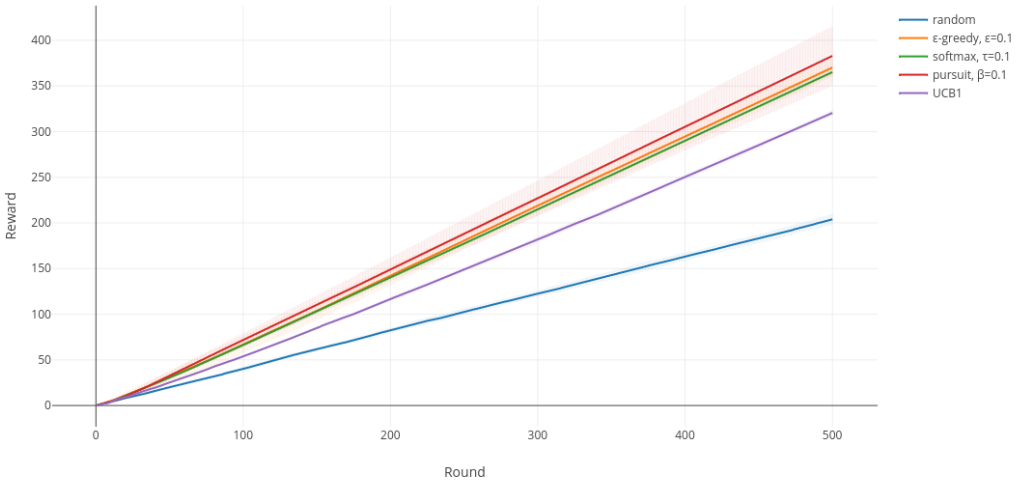

Figure 15: Cumulative rewards per round - 3C experiment

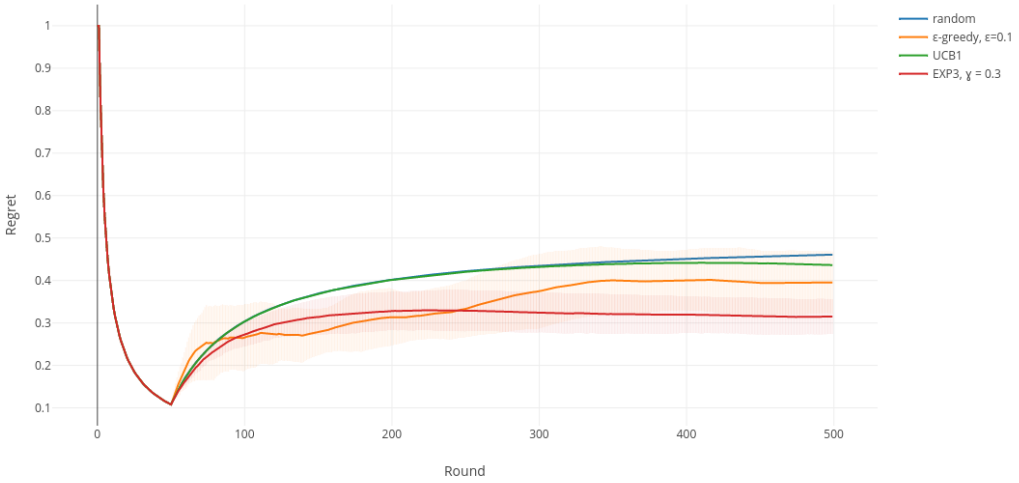

Figure 16: Regret per round - 4A experiment

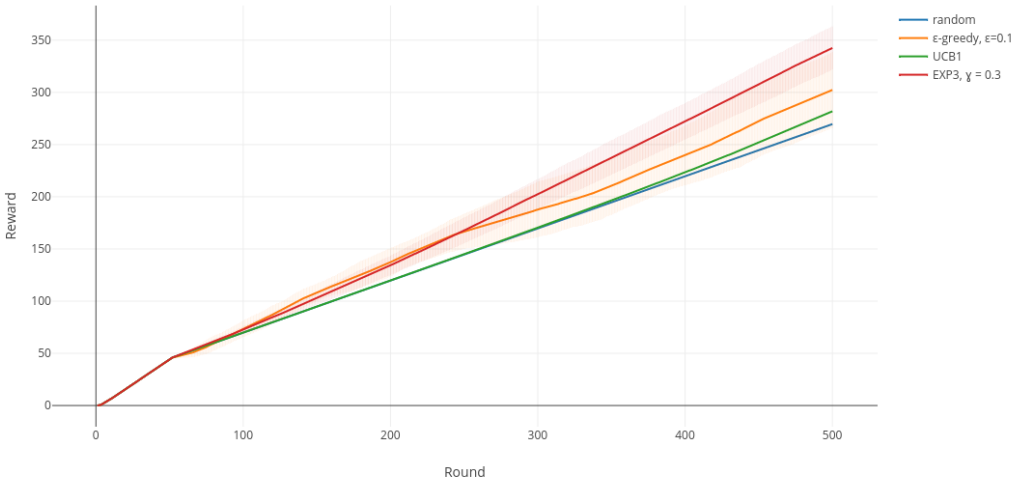

Figure 17: Cumulative rewards per round - 4A experiment

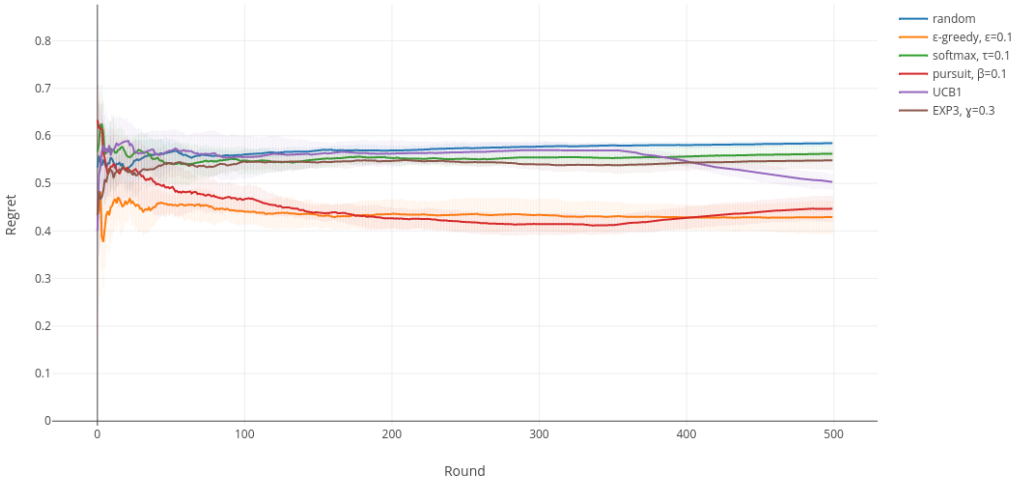

Figure 18: Regret per round - 5A experiment

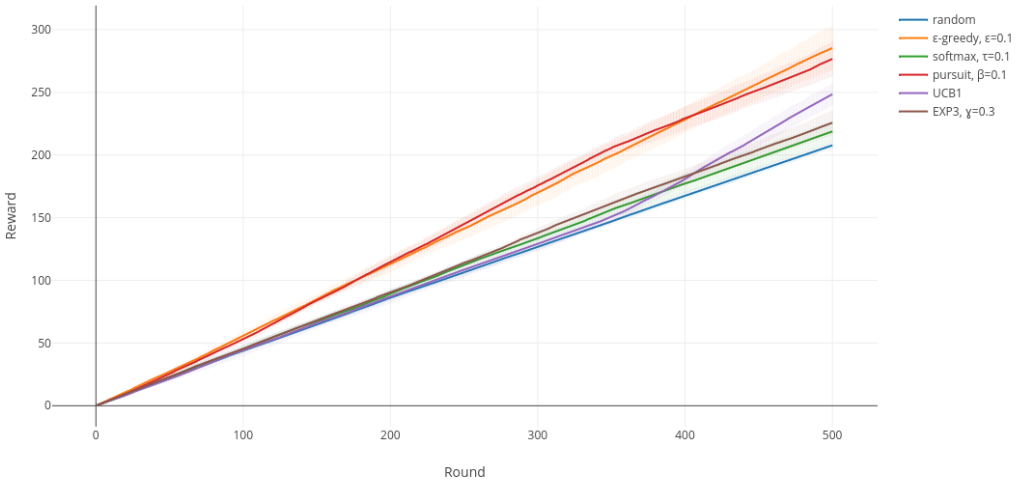

Figure 19: Cumulative rewards per round - 5A experiment

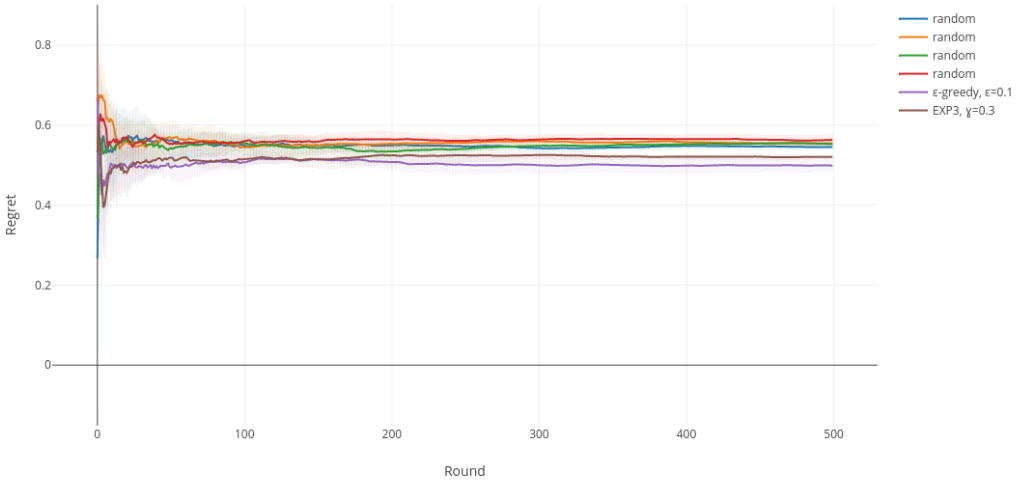

Figure 20: Regret per round - 5B experiment

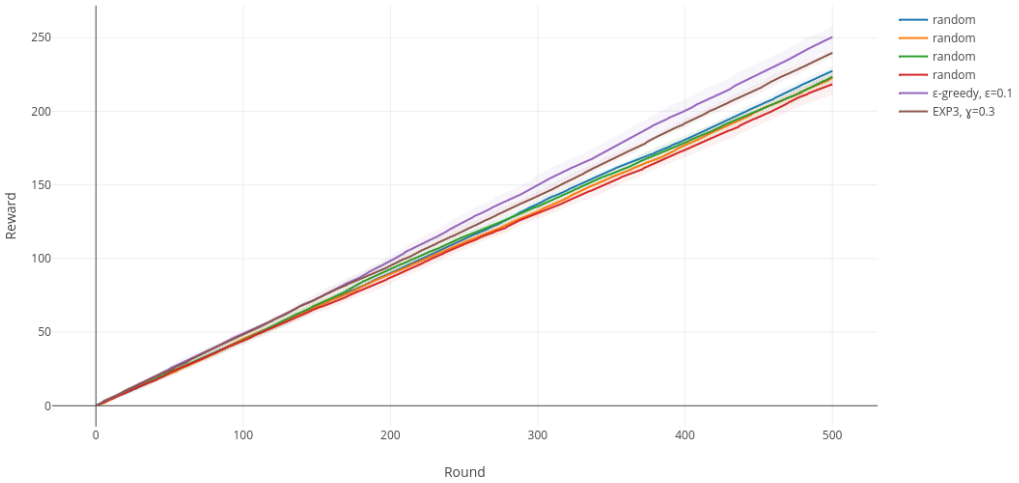

Figure 21: Cumulative rewards per round - 5B experiment

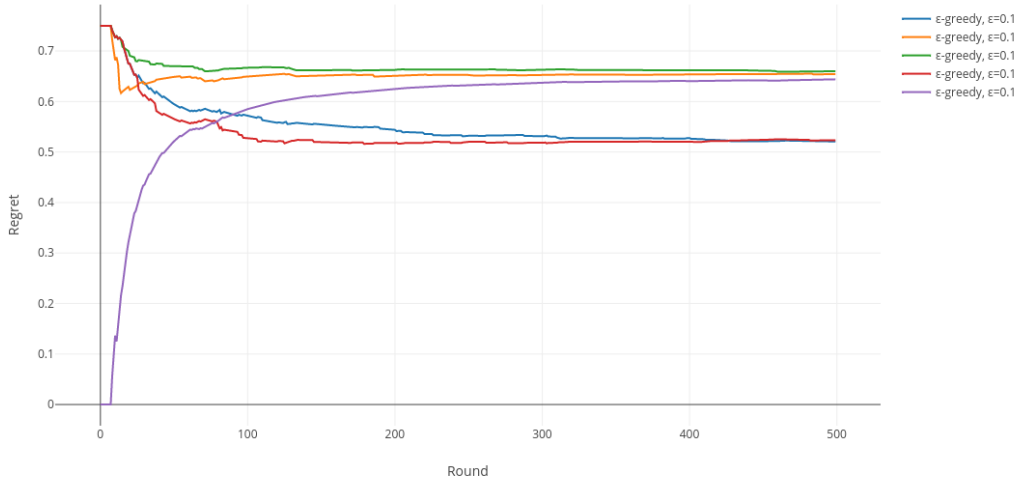

Figure 22: Regret per round - 5C experiment

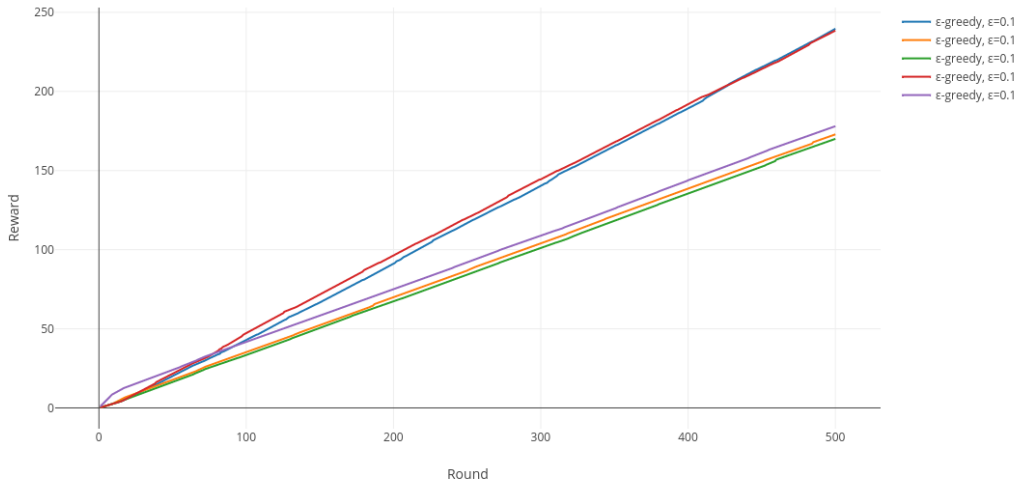

Figure 23: Cumulative rewards per round - 5C experiment

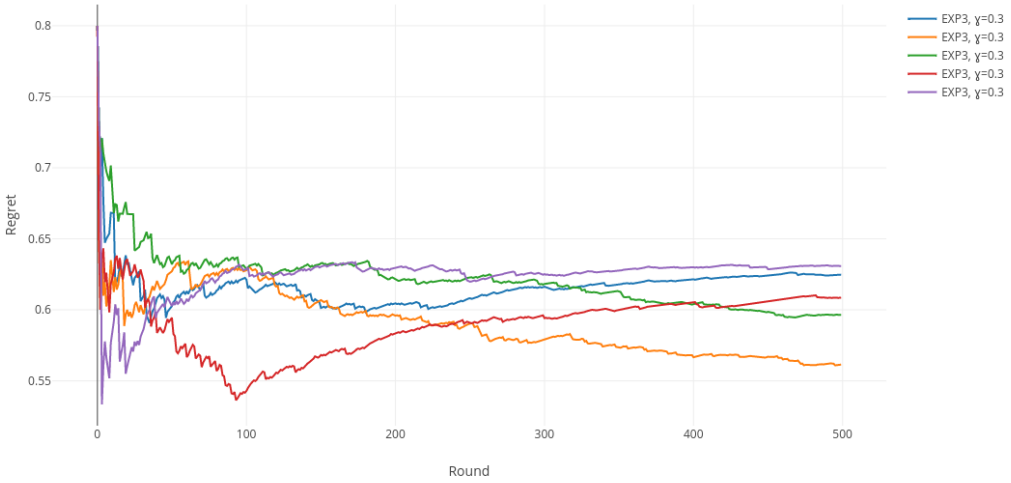

Figure 24: Regret per round - 5D experiment

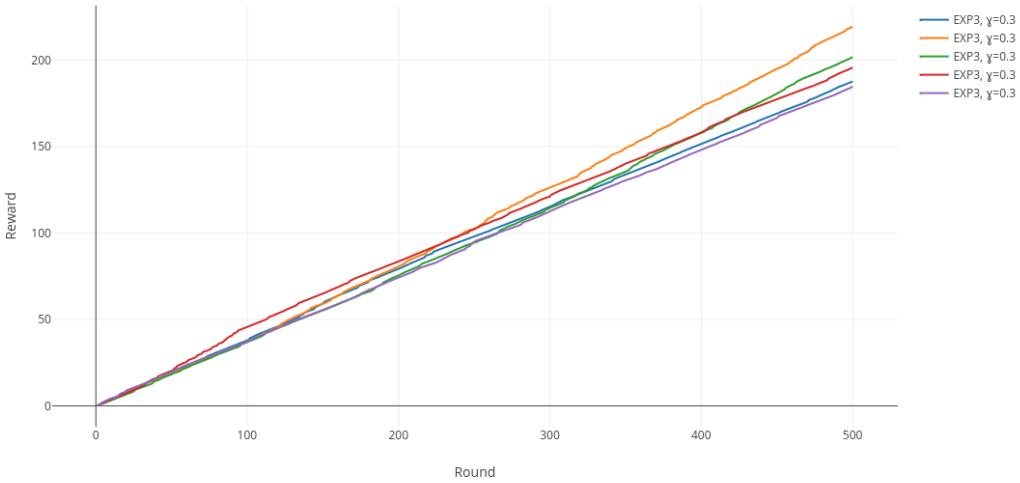

Figure 25: Cumulative rewards per round - 5D experiment

5 Future Work

First of all, the evaluation platform can be improved by offering a larger set of both arms and agents : various bandit algorithms not covered in the project could be implemented (e.g., UCB1-Tuned or EXP4) [7].

With regard to the traffic-like model used this the project, it seems important to study a different way of design this arm, by making it more *realistic*. One idea is to take in consideration the order of agent arrivals in the reward process for one round : an agent that requests an arm i at time t will get a better reward than those who requested the same arm i at time t'. The platform architecture can already support this improvement, at the level of the simulation node which can save the order of agent arrivals before transferring to arms.

Another improvement can be done in measure performance tools. For example, the simulation node should compute the percentage of rounds in which an agent pulls the optimal arm, that could help us to know more about the behaviour of bandit algorithms.

Most of all, future work will consist in applying other experiments. We did not consider the impact of the crossing between different models (e.g. to cross adversarial arms with stochastic arms), that might change the making decision process.

6 Conclusion

In this project, we have experimentally studied a set of multi-armed bandit algorithms allowed by the implementation of a modular evaluation platform. Its architecture enables parallel processing and multi-agents modeling.

A set of multi-armed bandit strategies were implemented to compare them in various situations. Our goals were to find aspects of a bandit that affect the regret of a strategy, and then measure how precisely they affect the performance. Number of arms and reward variance can influence the efficiency of a bandit policy. Then, by comparing strategies together, Pursuit bandits work well in stochastic models, while in adversarial environments, EXP3 agents become leaders. When a problem tends to be *real world* like, especially in a multi-agents simulation, it is hard to say which family of algorithms suits better. It depends on the way rewards are given, but also on the behaviour and choices of other agents that have the same goal : to minimize regret and maximize profit.

To transcend the limits imposed by multi-agent systems, we can think of a collaborative bandit environment [8] where agents have the opportunity to communicate together to make reliable decisions.

References

[1] T. Lai and H. Robbins, "Asymptotically efficient adaptive allocation rules," *Advances in Applied Mathematics*, vol. 6, no. 1, pp. 4 – 22, 1985. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0196885885900028

[2] S. Bubeck, N. Cesa-Bianchi *et al.*, "Regret analysis of stochastic and nonstochastic multi-armed bandit problems," *Foundations and Trends® in Machine Learning*, vol. 5, no. 1, pp. 1–122, 2012.

[3] W. contributors, "Multi-armed bandit — wikipedia, the free encyclopedia," 2018, [Online; accessed 12-January-2018]. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Multi-armed_bandit&oldid=819928466

[4] J. Kun. (2013) Optimism in the face of uncertainty: the ucb1 algorithm. [Online]. Available: https://jeremykun.com/2013/10/28/optimism-in-the-face-of-uncertainty-the-ucb1-algorithm/

[5] C. Stucchio. (2014) The adversarial bandit is not a statistics problem. [Online]. Available: https://www.chrisstucchio.com/blog/2014/adversarial_bandit_is_not_statistics_problem.html

[6] V. Kuleshov and D. Precup, "Algorithms for multi-armed bandit problems," *CoRR*, vol. abs/1402.6028, 2014. [Online]. Available: http://arxiv.org/abs/1402.6028

[7] C. Szepesvari. (2016) Contextual bandits and the exp4 algorithm. [Online]. Available: http://banditalgs.com/2016/10/14/exp4/

[8] R. K. Kolla, K. Jagannathan, and A. Gopalan, "Collaborative learning of stochastic bandits over a social network," in *Communication, Control, and Computing (Allerton), 2016 54th Annual Allerton Conference on.* IEEE, 2016, pp. 1228–1235.