Semi-bandits for Decentralized Optimal Transport on a Graph

AL Houceine KILANI
Ecole Normale supérieure Paris-Saclay
houcinekila@gmail.com

Salmane NAOUMI
Ecole Normale supérieure Paris-Saclay
salmane.naoumi@gmail.com

Julien SEZNEC
lelivrescolaire.fr
julien.seznec@lelivrescolaire.fr

Abstract

In this project, we consider a mass (e.g. people) transporting on a resistive graph. From a global perspective, a central planner might be interested in choosing each route to minimize the global cost (quadratic in each edge flow). The constraint that each unit of mass starts from its starting node and successfully reaches its endpoint can be formulated as a linear constraint. Hence, this is a standard convex optimization problem. Nonetheless, in a decentralized setting, each player will choose its route to minimize its local cost (linear in the flow of each visited edge). To do so, players are not allowed to communicate and are hence doomed to try routes sequentially. Moreover, at each try, all the players choose a route and only observe the flow in the visited edges (semi-bandit feedback). The cost of a route is therefore dependent on the other players actions. Hence, rewards are not stochastic, but probably not fully adversarial as well.

1 Introduction

In this project, we place ourselves in an Online Learning setting. Online learning is a class of machine learning problems where data becomes available sequentially and is used to update the best predictor for future data at each step. This is in contrast to the general approach towards learning problems where the best generator is learned from the entire training dataset at once.

In the online learning environment, an agent has a selection of options that they can choose from. Once an option is selected the environment provides some cost or reward feedback. The feedback may provide information on all of the available options (i.e., full-information), on only the selected options (semi-bandit) or perhaps only the aggregate of the chosen options (full-bandit). The player then has a decision in the next round, whether to continue with the option chosen in the previous round or to try/explore something different. It is usually a good idea to explore the dierent options available before exploiting the ones which appear to be the best at the current level of information. There has recently been a lot of interest in developing algorithmic strategies for addressing this exploration vs exploitation trade-off to achieve most ecient results.

2 Scope of our work

The environments we model are road/traffic networks. Such a network can be expressed as a graph $G = (V, E)$ where V is the set of vertices and E the set of edges. The vertices represent points on the network, and links, the roads between those points. Every road has a cost, and if too many masses are on that edge, the more congested it is and the more will its cost be. For example, they can represent travellers between metro stations and the cost paid is the suffering from the train being crowded. A route on the network is a series of edges connecting an origin and a destination. This route has a cost defined as the sum of the costs of each individual edge.

Since the goal of each agent will be to minimize its own cost of travel, we are inspecting how their choice of paths behave during the interactions on the network. For this we are going to place a number of agents using different Bandits strategies (see section 3), and study :

- Which of the semi bandits agents converge to an equilibrium ?
- How fast they converge ?
- What if there are perturbations on the network (players with no specific strategy) ?

3 Bandits used

3.1 Stochastic:

Stochastic combinatorial bandits under semi-bandit feedback consider that the sequence of costs $(C_i(n))_{n\geq 1}$ is i.i.d. w unknown but fixed. They are assumed to be independent across actions.

Theorem 1 *(Instance-dependent). For the Stochastic Combinatorial semi-bandit problem, any algorithm achieves a regret of at least [5]*

$$R(T) \geq O\left(K \times m \times \frac{1}{\Delta}\right) \tag{1}$$

Especially, the implemented algorithm CombUCB1 achieves a regret of at most

$$R(T) \leq O\left(K \times m \times \frac{1}{\Delta} \times \log T\right)$$

3.2 Adversarial:

Adversarial combinatorial problems under semi-bandit feedback consists of considering an adversarial environment where the adversary decides the loss of each arm, then the learner suffers and observes (only) the loss of the picked arm (costs of edges in the chosen path). The implemented algorithm is EXP2 [1], it satisfies for any concept class $S \subset \{0,1\}^d$ and any L_∞-adversary, the regret R_n satisfies :

$$R_n \leq 4d^2\sqrt{n} \tag{2}$$

3.3 Hybrid:

Hybrid semi bandits are algorithms that are able to deal simultaneously with stochastic and adversarial environments. For Example, we implemented a general semi-bandit algorithm based on the Follow-Perturbed-Leader (FPL) [4] for our problem setting that showed to achieve $\mathcal{O}(\log T)$ regret for stochastic environments and $\mathcal{O}(\sqrt{T})$ regret for adversarial environments without prior knowledge of the regime or the number of rounds T.

4 Environments used

As mentioned earlier, the environments are graphs $G = (V, E)$. Using the 'Build.py', we build one of the graphs below and return possible edges $E = (start, destination)$ between the nodes, their costs and an adjacency matrix (all needed to be used in the agents). In the subsections below, we present each graph.

4.1 Random sparse graph

Given $|V|$ the number of vertices, we construct a fully connected graph. Then we discard randomly a fraction r of the edges from the graph. There may be the case where we get more than one connected component, and thus there may not be a path between two given nodes. To tackle this, we check if there exists a road or not between the starting node and destination road randomly assigned to the agent.

This is a simple graph having $\forall e \in E, cost_{base}(e) = 1$ (in this case this corresponds to how many agents used that particular edge at the same time). The goal of using this graph is to flexibly change the number of vertices and thus control the collisions between the agents and study the behaviours of the agents accordingly (used for debugging also since the global optimum is computed quickly on this graph)

4.2 OW net

This network is extracted from Example 10.1 from the book "Ortúzar Willumsen (2011) Modelling transport". It is a graph of 13 Nodes (A to M) linked with multiple edges having different costs. It was first created to model traffic routes in a town center to identify set of routes which might be considered attractive to drivers. Diversion across routes in their case is due to capacity restraint. In our setting, this would be equivalent to amplifying agents costs depending on flow conditions. Therefore, The differences in perceptions and knowledge of semi-bandit algorithms would also lead to a spread of routes and to stochastic or adversarial elements in route choice.

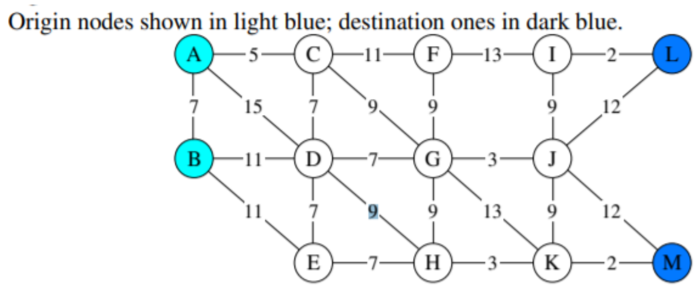

Origin nodes shown in light blue; destination ones in dark blue.
A
B
C
D
E
F
G
H
I
J
K
L
M
5
11
13
2
7
15
7
9
9
9
12
11
7
3
11
7
9
9
13
9
12
7
3
2

Figure 1: The OW network used

4.3 Sioux Falls graph

The Sioux-Falls network is a graph of 24 nodes and 76 links that was used in many publications for transportation and traffic assignment. However, it is not considered as a realistic one. . All network data including the link numbers indicated on the map (but excluding node coordinates), are taken from [3]. It is richer than the OW network (thus more time is needed to calculate the different paths on it) and has non trivial edge's costs (For exemple one can prefer taking and edge with other agents than taking another high base cost edge). Plus we saw that there were some slight differences in the costs of edges given the direction but at it was very slight we did not pay attention to it and considered it as they were equal.

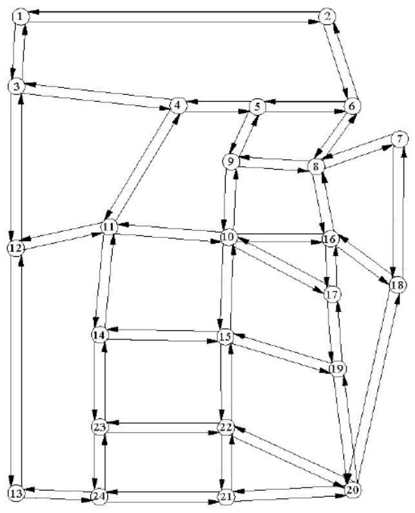


Figure 2: The Sioux Falls network

5 Analysis of the interactions

The simulation protocol corresponds to the following :

- Build the graph
- $\forall$ a $\in$ agents, Define s_a the starting node and d_a the arrival node
- $\forall$ a $\in$ agents, Compute the set P^a_{sd} representing all the possible paths between s_a the starting node of agent a and d_a the arrival node of agent a
- for $t = 1..T$
  - $\forall$ a $\in$ agents, $arm_a \sim \mathbf{P}_t(.|(strategy_a, P^a_{sd}))$ where $strategy_a \in [stochastic, adversarial, hybrid]$. In this case $arm_a \in P^a_{sd}$
  - Observe real costs $C_t(e)$:

$$C_t(e) = \sum_{a \in agents} \sum_{e \in E} 1(e \in arm_a) * cost_{base}(e)$$

  - Update $\mathbf{P}_{t+1}(.|(strategy_a, P^a_{sd}))$

For the random sparse graph, we adjust r according to the number of vertices wanted so as to get a reasonable size for P^a_{sd}. Keeping $|V|$ low also helps to limit the number of possible paths but also increases the chances of "collisions" between the playing agents. When we observe a quick convergence of the agents, we increase their number (or lower the number of vertices in the Random Sparse Graph) so as to increase the number of collisions. In order to observe some behaviours (Decreasing cost of adversarial agents for instance -see below-) we may vary the number of iterations that was set by default to $T = 1000$

In each time we study the convergence of the agents (or not) and compare the "final" strategies and costs with the ones provided by the Oracle agent **(given the graph, the agents and their paths, uses brute force to find the total minimum cost on the network, but since it is a combinatorial problem, we put a time limit thus approximating the optimal cost - this is why it may happen that we see Oracle's optimal cost higher than found costs, to remediate to this, just increase the searching time limit-)**

5.1 General observations :

In all the graphs, we observe that the agents that look at the costs in an hybrid way are the ones that perform poorly. The stochastic agents are the one that converge to a stable choice in a number of iterations increasing with the exploration parameter of UCB1 algorithm. The hybrid agents go through a transitional phase where their cost is high but it decreases afterwards. Both the Hybrid and

Adversarial agents keep oscillating between a set of arms, nevertheless the variance of the costs for the adversarial agents stays high relatively to the one of the hybrid agent.

We fixed the number of iteration to $T = 1000$ but decrease it when we see that there is no need to go further sometimes. We choose $\gamma = 1$ for the hybrid agents and $\eta_t = \frac{1}{\sqrt{t}}$ for the adversarial agents

When we mention ‘relative difference‘, this means the quantity :

$$d = \frac{|o - \hat{o}|}{o}$$

where o is the optimal total cost found by the oracle and $\hat{o}$ is the total of the costs found by the agents.

5.2 Behaviour of the agents in presence of the other’s strategies

See the jupyter notebooks corresponding to the analysis for complete figures and details on the paths obtained.

5.2.1 One strategy

When there are only stochastic agents on the network, we observe that they quickly converge to choosing an arm and stay using it over and over. Depending on how we fix the exploration parameter, this maybe be the optimal arms (Random Graph) or a very near optimal one (OW and Sioux Falls).

Running one adversarial agent only in the environment or a small number of agents that yields to a low chance of collision does not make sense since the environment’s costs wont be acting like the adversarial hypothesis suppose. Running 5 adversarial agents show that these agents converge to the choice of sub optimal arms (relative difference of 47% between the optimal found by the agents and the one given by the oracle). Plus the variance is high and it keep fluctuating over and over between the arms.

Hybrid agents behave a bit like adversarial ones in term of convergence (a transitional phase where the costs are decreasing) and then they stabilize around some near optimal arms. But here we observe that the cost variance is low but the optimum found is highly suboptimal (100% relative difference).

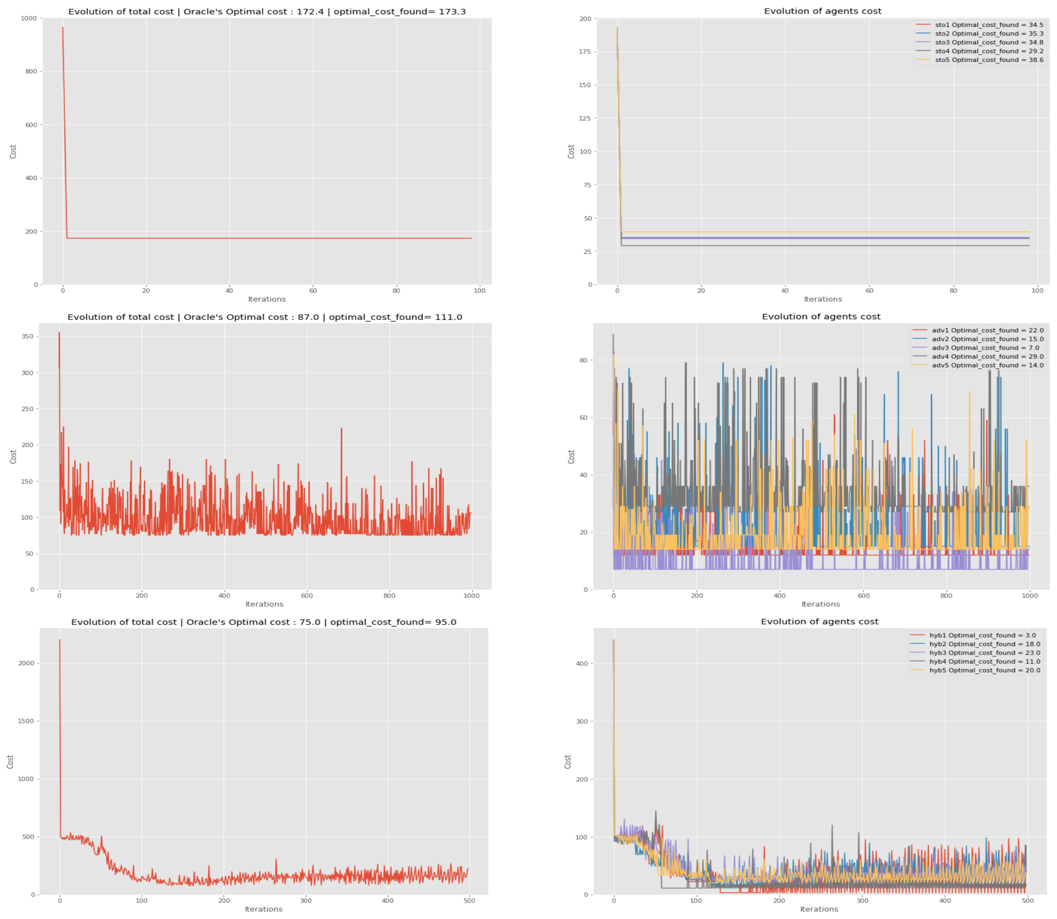


Figure 3: 1 strategy on the OW network

5.2.2 Two strategies

Making a Stochastic vs Adversarial play shows the same behavior as when all the players interacting are using the same strategy. Stochastic agents converge faster towards choosing arms not necessarily optimal ones but pretty close to them (6% relative difference). Same thing for the adversarial agents with a high cost variance. The interesting thing we saw here, concerning the paths chosen, is that the stochastic agents acting selfishly (stochastic agent number 5 picking a low cost arm for himself) impact the other adversarial in the way that he pushes them to pick slightly high costs arms increasing the overall cost. But the arms picked by adversarial remained close to the optimal ones nonetheless and the optimal ones may be reached if we give them enough time to search (as for the OW graph figures -see github for more figures-)

Same thing goes for the stochastic vs hybrid play their behavior remained the same as before but this time with the hybrid agents having a more moderate variance of the costs.

Making hybrid agents play with the adversarial agents however shows an interesting behaviour. Hybrid agents prefer taking longer paths to avoid collisions with other players which leads to worsening their individual costs and the variance of the arm on which they oscillate increases.

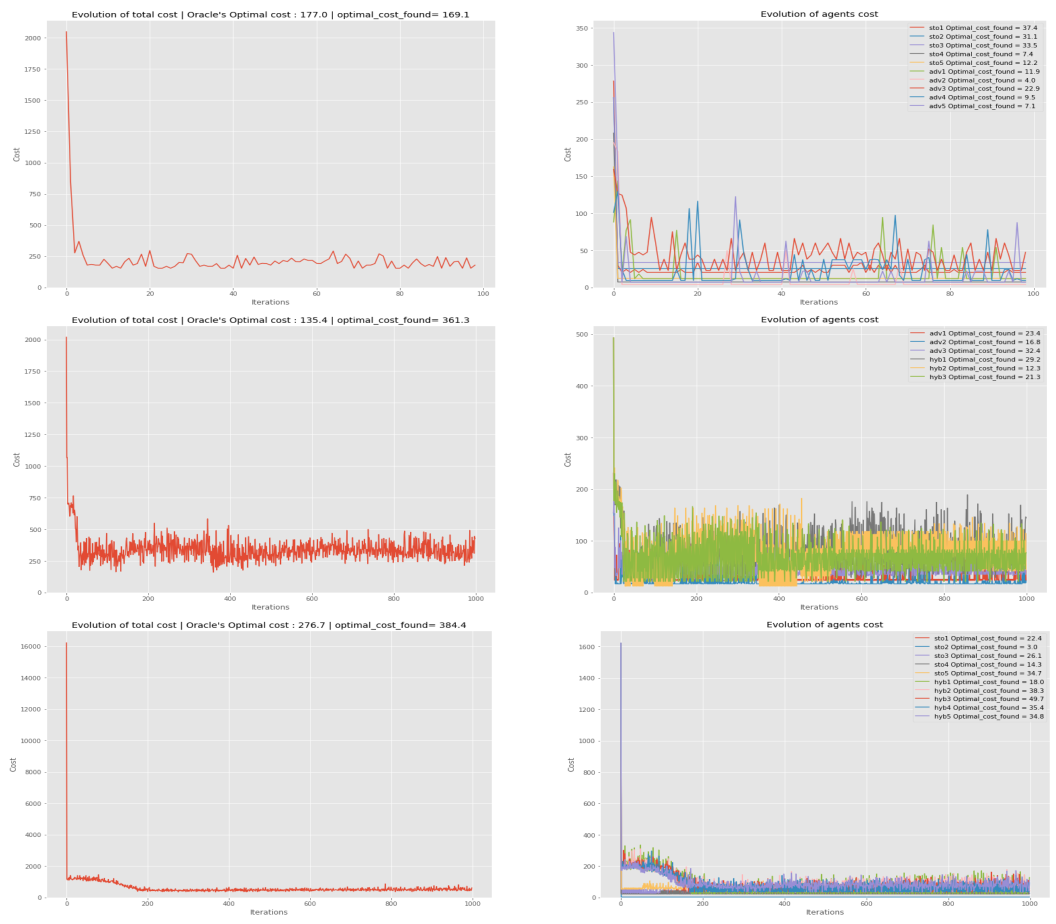


Figure 4: 2 strategies on the Sioux Falls network

5.2.3 Three strategies

Having the three agents act on the same graph does not change a lot their pairwise behaviors mentioned earlier : Stochastic agents find an optimum rather quickly, followed by the adversarial then the hybrid but the two latter keep oscillating between arms and do not find an equilibrium yielding to an unstable overall found cost and high cost variance for the hybrid agent seen costs. The fact that the stochastic agents act selfishly thus impacting the others choice to take longer paths is being seen here too for the Sioux Falls graph.

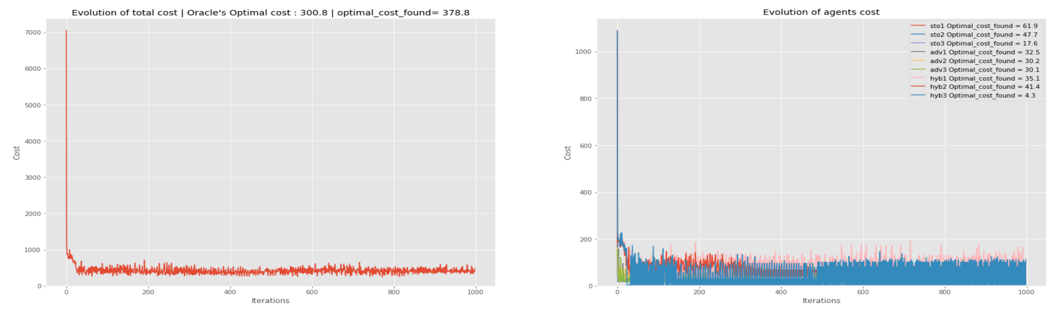


Figure 5: 3 strategies on the Sioux Falls network

5.3 Varying the parameters of algorithms

5.3.1 Stochastic strategy

The way of controlling the exploration regime in CombUCB1 is the exploration parameter. When Varying this parameter in the case of environments with no random or adversarial agents, we observe no difference in the behavior of the algorithm. However, once there exists some other noisy agents in the graph, using a lower exploration parameter leads to an increase in the variance of the total cost of the agent. Thus a good value for the exploration parameter is 5 (the parameter used for the previous experiments about the interactions of the strategies). The results found are the following :

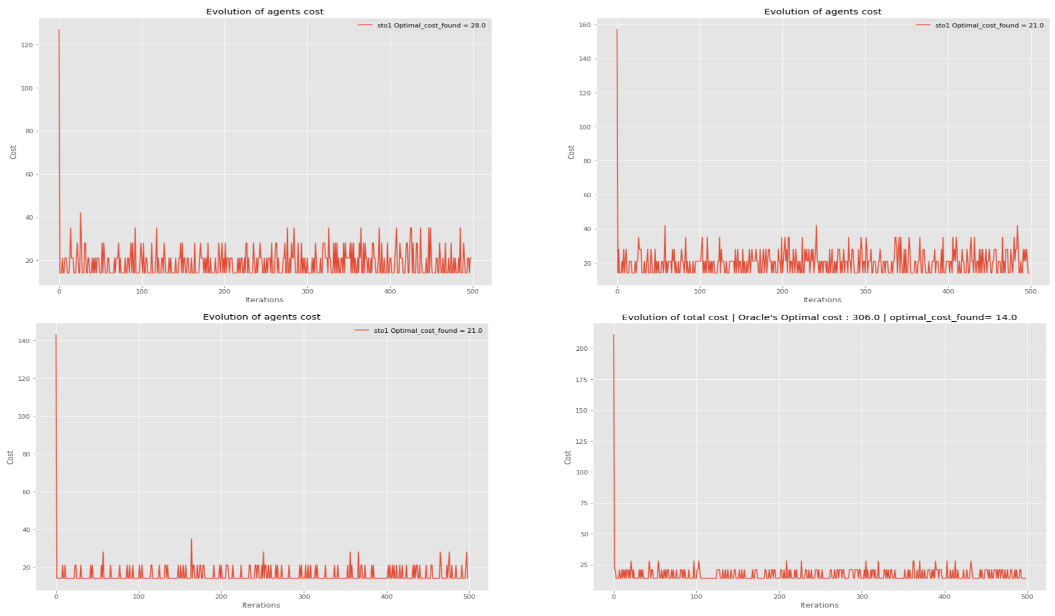


Figure 6: Impact of exploration parameter on the evolution of total cost of stochastic agents

5.3.2 Adversarial strategy

For the Exp2 algorithm, the only parameter we can change is the learning rate η_t. In fact, different learning rate schemes has lead us to different behaviors. We used 5 different functions for computing a learning rate at each iteration. $1/\sqrt{t}, 1/(4\sqrt{t})$, $4\sqrt{\log(t)/t}$ and 0.1.

The results below show that $1/(4\sqrt{t})$ appears to be efficient and stable over other learning rate strategies.

5.3.3 Hybrid strategy

As for the previous strategies, FPL algorithm showed a change in behavior w.r.t the exploration parameter. In fact, the results are showing a huge variance in the total cost of environments with hybrid agents. Moreover, increasing exploration parameter γ implied a decrease in variance and the total cost but more iterations seemed to be necessary to reach the equilibrium.

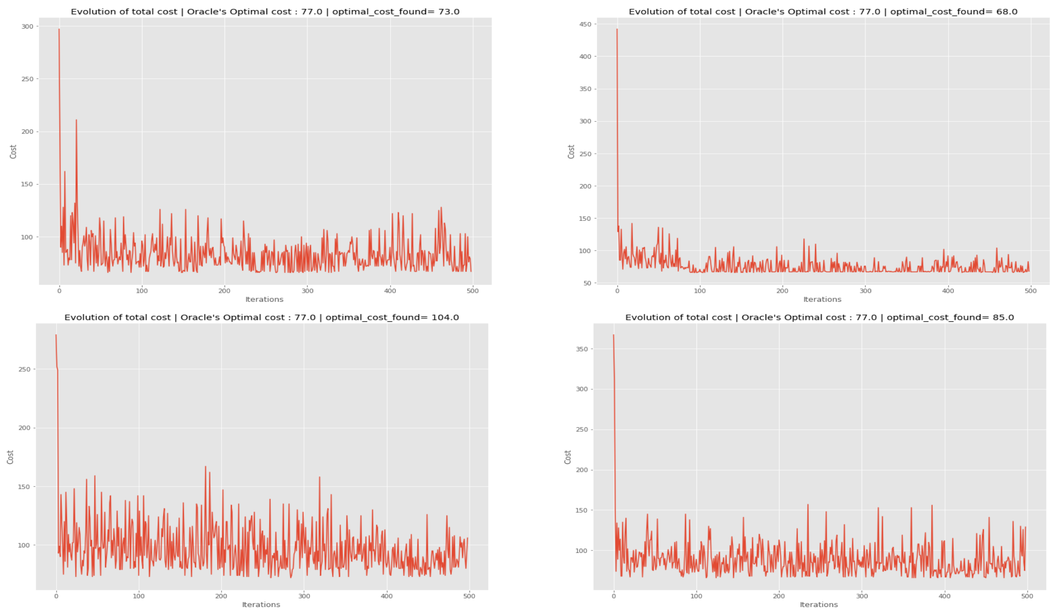


Figure 7: Impact of Learning rate strategy on the evolution of total cost in an environment with adversarial agents

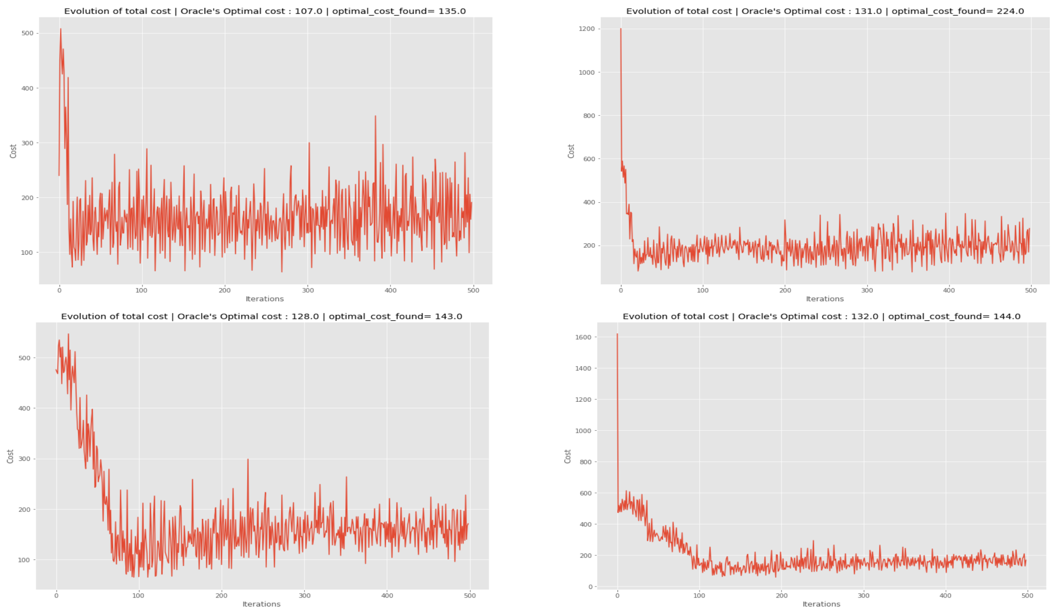


Figure 8: Impact of exploration parameter γ on the evolution of total cost in an environment with Hybrid agents

5.4 Varying the number of agents

In a pure 1 strategy play (Only stochastics, only adversarials, only hybrid), increasing the number of players makes the collisions happen more frequently but this does not change the behaviour of convergence or the variance mentioned before. But we see that the more agents there are :

- in the stochastic play, the further away is the convergence towards the final chosen arms and the total cost in not that close from the global optima (13% relative difference).
- in the hybrid play, the relative difference to the Oracle's optimum decreases (5 agents : 400%, 8 agents : 120%, 10 agents : 80%).

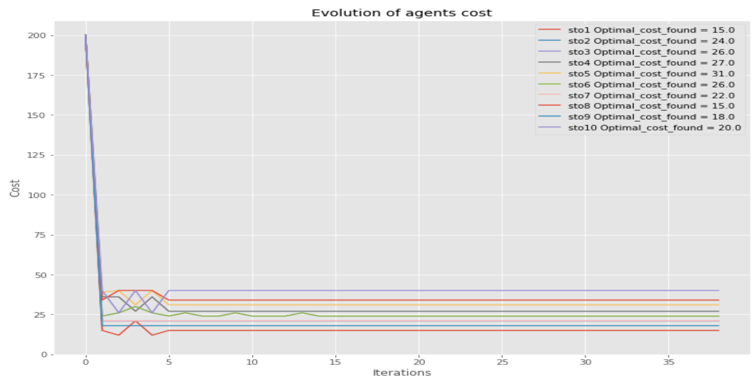
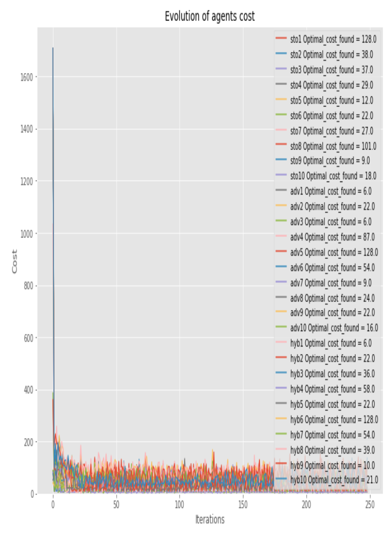


Figure 9: 10 stochastic agents on the OW network

Same thing in a mixed play, The behaviours remain the same with the selfish behaviour of the stochastic agents observed in the pairwise interactions. Indeed here we see for instance, the stochastic agents 1, 2, 8 take paths whose costs are lower than the ones assigned by the Oracle, and since they are the stochastic agents are the one that converge faster and do not change their choices, they force the other agents to pick high cost paths, inducing then an increase in the global cost.



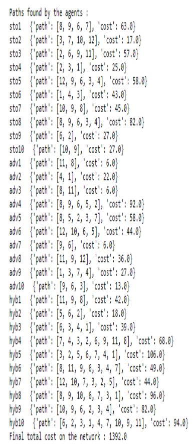

```
Paths found by the agents :
sto1   {'path': [8, 9, 6, 7], 'cost': 63.0}
sto2   {'path': [3, 7, 10, 12], 'cost': 17.0}
sto3   {'path': [2, 6, 9, 11], 'cost': 57.0}
sto4   {'path': [2, 3, 1], 'cost': 25.0}
sto5   {'path': [12, 9, 6, 3, 4], 'cost': 58.0}
sto6   {'path': [1, 4, 3], 'cost': 43.0}
sto7   {'path': [10, 9, 8], 'cost': 45.0}
sto8   {'path': [8, 9, 6, 3, 4], 'cost': 82.0}
sto9   {'path': [6, 2], 'cost': 27.0}
sto10   {'path': [10, 9], 'cost': 27.0}
adv1   {'path': [11, 8], 'cost': 6.0}
adv2   {'path': [4, 1], 'cost': 22.0}
adv3   {'path': [8, 11], 'cost': 6.0}
adv4   {'path': [8, 9, 6, 5, 2], 'cost': 92.0}
adv5   {'path': [8, 5, 2, 3, 7], 'cost': 58.0}
adv6   {'path': [12, 10, 6, 5], 'cost': 44.0}
adv7   {'path': [9, 6], 'cost': 6.0}
adv8   {'path': [11, 9, 12], 'cost': 36.0}
adv9   {'path': [1, 3, 7, 4], 'cost': 27.0}
adv10   {'path': [9, 6, 3], 'cost': 13.0}
hyb1   {'path': [11, 9, 8], 'cost': 42.0}
hyb2   {'path': [5, 6, 2], 'cost': 18.0}
hyb3   {'path': [6, 3, 4, 1], 'cost': 39.0}
hyb4   {'path': [7, 4, 3, 2, 6, 9, 11, 8], 'cost': 68.0}
hyb5   {'path': [3, 2, 5, 6, 7, 4, 1], 'cost': 106.0}
hyb6   {'path': [8, 11, 9, 6, 3, 4, 7], 'cost': 49.0}
hyb7   {'path': [12, 10, 7, 3, 2, 5], 'cost': 44.0}
hyb8   {'path': [8, 9, 10, 6, 7, 3, 1], 'cost': 96.0}
hyb9   {'path': [10, 9, 6, 2, 3, 4], 'cost': 82.0}
hyb10   {'path': [6, 2, 3, 1, 4, 7, 10, 9, 11], 'cost': 94.0}
Final total cost on the network : 1392.0
```

Figure 10: Example of the "selfish" behaviour of stochastic agents

5.5 Robustness to noisy random agents

In this setting, the goal is to see the how the agents interacting with a noisy agent choosing paths randomly adapt to it and find their own optimal paths. Thus the focus will not be on the value of global optimal cost (since the noisy agent will never stop picking randomly) but rather on the behaviour of the agents (convergence and fluctuations of their own costs).

We observe that the more noise there is, the more perturbed are the stochastic agents as well as the hybrid one. Indeed, we see that the stochastic agent fluctuates between the arms he chooses a lot and needs a lot of iterations before the variance of the cost of the paths he chooses decreases. The hybrid agent, the same way as before, picks less costly paths but then performs an oscillation between the arms keeping the cost's variance really high and getting worse and worse as the iterations go. However the adversarial agent does not seems to be impacted at all by the noise since it main assumption if that the environment plays against him by choosing a set of costs/rewards at each iteration which is kind of exactly the case here. It is under these settings that we see that the cost of adversarial agents chosen path are have the lowest cost mean.

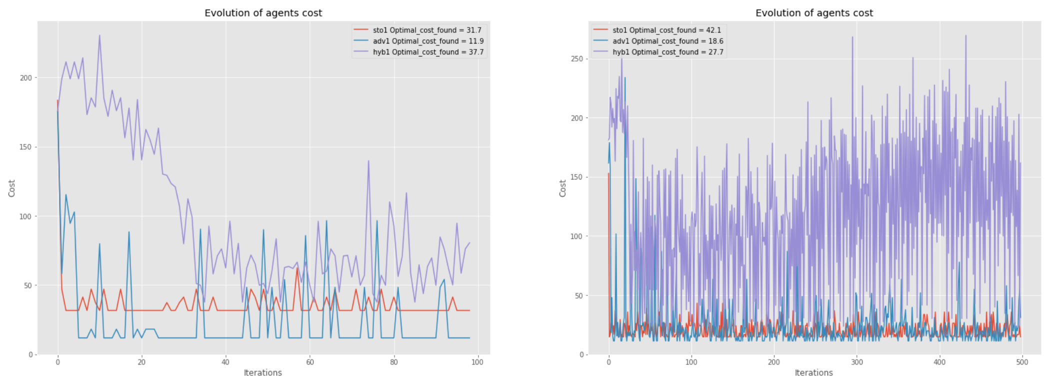


Figure 11: 1 noisy agent playing (left) vs 10 noisy agents playing (right)

6 Conclusions :

The work discussed here was to see what would happen in a mix congestion play (on a network) between Stochastic agents (running on UCB1 algorithm), Adversarial agents (Exp2 algorithm[1]) and Hybrid agents (FPL-Trix algorithm [4]). Given the experiments on different networks, we showed that there are advantages and disadvantages to each of the agents depending on the setting. For instance the stochastic agents (with a good exploration parameter) perform very well by finding the optimal paths faster. But they show a selfish behaviour (quickly picking some low cost arms and never moving out from them) that forces the others agents to pick long paths and thus increasing the overall cost. They also fail when presented in an environment with an important number of noisy agents interacting on it and in a mixed play.

In contrary, the adversarial agent are not impacted by these noisy agents or by any other agent playing simultaneously on the network but they suffer from oscillation between paths (even though these paths are close to the optimal ones).

The hybrid agents presented a very interesting behaviour where they start like the stochastic agents (picking high costs arms) but switch to picking lower costs ones after a sufficient amount of iterations. But their principal inconvenient was being very sensitive to the agents that he cannot predict their next moves (adversarial/noisy) leading it to have a very high variance of the costs of the arms he picks.

The convergence towards the global optimum was only observed in 1 strategy settings (there is a need to increase the number of iterations to observe it for adversarial and hybrid agents) and in some runs of a mixed play between stochastic and adversarial plays. Outside of the settings, the convergence happening was not towards global optimums but rather to multiple local ones (oscillation).

Other interesting questions were unveiled during this work such as what can be done to find a better global optimum in this setting rather than using our brute force solution ? how to limit the oscillations of the adversarial and hybrid agents when both present in the congestion game ? Also maybe trying other algorithms to confirm the general behaviours observed and guarantee that it is not due to the algorithms used (Stochastic USCB and adversarial CombExp [2] or GeoCombUCB [8] or LogBarrier [6] and Hybrid FTRL [7] . Can we modify the stochastic agents in a way such that it forces them to get out our their selfish pick and benefit the whole game lowering the total cost ? (increasing the exploration parameter was a solution but we then loose its fast convergence property). We hope this work to be a starting point for answering such questions and allowing to have a better explanation for these mixed plays that do not fall neither in the fully stochastic settings nor in the fully adversarial ones.

References

[1] Jean-Yves Audibert, Sébastien Bubeck, and Gábor Lugosi. “Regret in Online Combinatorial Optimization”. In: *CoRR* abs/1204.4710 (2012). arXiv: `1204.4710`. URL: `http://arxiv.org/abs/1204.4710`.

[2] Richard Combes et al. “Stochastic and Adversarial Combinatorial Bandits”. In: *CoRR* abs/1502.03475 (2015). arXiv: `1502.03475`. URL: `http://arxiv.org/abs/1502.03475`.

[3] Larry J. LeBlanc, Edward K. Morlok, and William P. Pierskalla. "AN EFFICIENT APPROACH TO SOLVING THE ROAD NETWORK EQUILIBRIUM TRAFFIC ASSIGNMENT PROBLEM. IN: THE AUTOMOBILE". In: Transportation Research Vol. 9, pp. 309-318, 1975 (1975).

[4] Gergely Neu. "First-order regret bounds for combinatorial semi-bandits". In: *CoRR* abs/1502.06354 (2015). arXiv: `1502.06354`. URL: `http://arxiv.org/abs/1502.06354`.

[5] Karthik Abinav Sankararaman. "Semi-Bandit feedback: A survey of results". In: (2016). URL: `http://karthikabinavs.xyz/surveySemiBandit.pdf`.

[6] Chen-Yu Wei and Haipeng Luo. "More Adaptive Algorithms for Adversarial Bandits". In: *CoRR* abs/1801.03265 (2018). arXiv: `1801.03265`. URL: `http://arxiv.org/abs/1801.03265`.

[7] Julian Zimmert, Haipeng Luo, and Chen-Yu Wei. "Beating Stochastic and Adversarial Semi-bandits Optimally and Simultaneously". In: *CoRR* abs/1901.08779 (2019). arXiv: `1901.08779`. URL: `http://arxiv.org/abs/1901.08779`.

[8] Zhenhua Zou, Alexandre Proutière, and Mikael Johansson. "Online Shortest Path Routing: The Value of Information". In: *CoRR* abs/1309.7367 (2013). arXiv: `1309.7367`. URL: `http://arxiv.org/abs/1309.7367`.