Factors that Influence Outcomes of Repeated Congestion Games

CHARLIE WU, CINDY LE, HAN LIU

1 INTRODUCTION

Repeated game is an interesting field in the game theory where agents play multiple rounds of a same game. Thus, the agents can either maintain or change their beliefs on actions of other agents based on historical outcomes. Agents also need to keep in mind that their actions will impact how other agents behave in the future. Repeated games allow agents to use learning algorithms like the fictitious play, or multi-armed bandit algorithms.

Repeated congestion game is a specific type of repeated game where agents are allowed to repeatedly play a congestion game. In this paper, we will explore various learning algorithms with distinct parameters in repeated congestion games and study how different factors may influence the outcomes of the game.

2 IMPLEMENTATION

2.1 File Structure

The Python program consists of a series of classes, a configure file, and a main file. Each road has a function that returns a cost of itself. Each agent is able to choose a path. Then, the main file puts everything together to run with certain agents and roads based on the configure file.

2.2 Agent Algorithms

Each agent may adopt an algorithm on the following list to decide which path to take in each round. They store the historical results if needed.

2.2.1 Fictitious Play. Each agent maintains an empirical probability distribution of paths that other agents choose, s_{-i}, and computes her expected payoff of each action $s_i \in S_i$ given her maintained empirical distribution of other agents' actions. Then she picks the action that yields the maximum payoff.

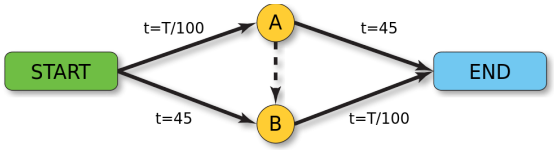
2.2.2 ε-greedy. Each agent has ε probability of choosing a random one among all possible paths. Otherwise, choose the one that have generated the best average outcome in history.

2.2.3 UCB1. Each agent chooses a path that maximizes $\overline{x_j} + \sqrt{\frac{2\ln n}{n_j}}$ [1], where $\overline{x_j}$ is the average reward, n_j is the number of tries for that particular path, and n is the number of total moves.

2.2 Road Configurations

We construct our roads according to the Braess Paradox. As shown below [2], there are four roads with well-defined cost functions, and 4000 agents. Before the zero-cost super highway is built between A and B, at the Nash equilibrium, equal amount of agents travel through Start - A -

End and Start - B - End. It costs 65 for each agent. There's no incentive for any agent to switch since if he/she switches the route, the cost will rise to 66. After the super fast highway is built, Start - A - B - End becomes a new Nash equilibrium route which costs 80 now. No agent, however, has incentive to switch, as the other routes Start - A - End and Start - B - End both cost 85. When adding the super highway that seems beneficial to everyone, agents actually spend a larger total cost travelling from one place to another due to their self-interested property.



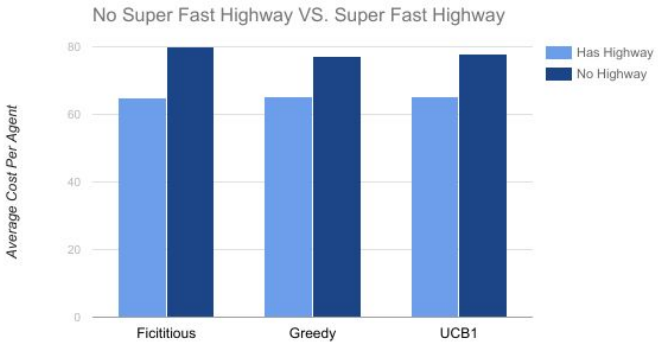
The special case may best illustrate the properties we are studying. To save computing resources, we change the cost function for Start - A and B - End from T/100 to T/1 in our implementation.

3 RESULTS AND DISCUSSION

In all experiments, we run 100 turns of simulations for data.

3.1 Super Fast Highway vs. No HighWay

Each run involves only one type of agent, and 40 agents are used per trial. Cost is averaged over 100 turns and over every agent type.

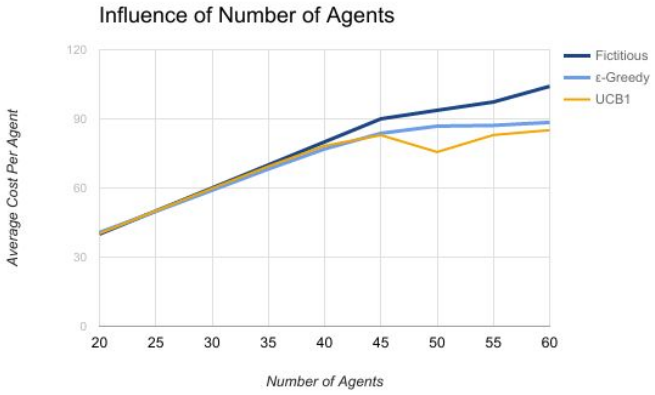


The super fast highway presents challenge for each agent type. The graph indicates that each agent experiences an increased cost after the super fast highway is added. The ε-greedy and the UCB1 algorithm consistently receive a relatively small penalty, presumably because the two

algorithms not only exploit the currently known best paths, but also explore new paths with a certain probability, so they do not that consistently stuck on the nash equilibrium path (also the worst case possible).

3.2 The Influence of Number of Agents for the Three Agent Types

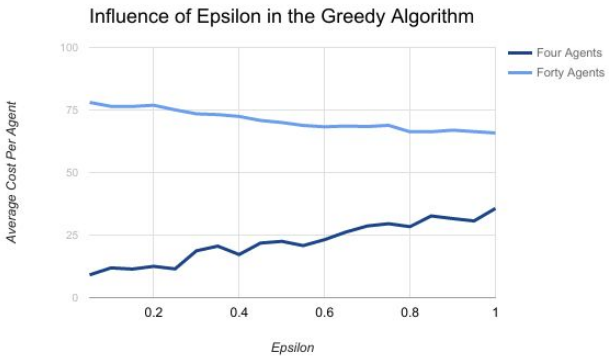
Each run only involves only one type of agent, and 40 agents are used per run. Cost is averaged over 100 turns and over every agent type.



30 agents is the turning point. With 30 agents, the Nash equilibrium case when all agents are on Start - A - B - End path has a cost of 60 per agent, while the case when half of the agents on Start - A - End and half on Start - B - End also has cost of 60 per agent. Above the turning point, the greedy algorithm gradually surpasses the fictitious algorithm because the Nash equilibrium case becomes the globally worst case. The exploration behavior in the ε-Greedy algorithm and the UCB1 algorithm explores the non-Nash equilibrium cases, and helps to reduce the cost.

3.3 The Influence of Epsilon in the ε-Greedy Algorithm

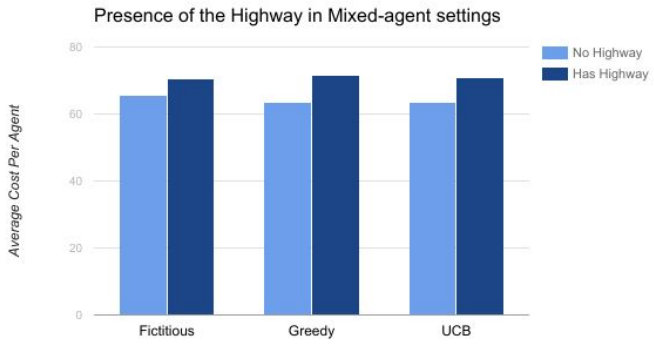
This experiment only involves the ε-Greedy agents.



An increased epsilon, or the probability to deviate from greedy strategy, would increase the cost in the four-agent case, but decrease the cost in the forty-agent case. When 4 agents are involved, the Nash equilibrium is that everyone stays on Start - A - B - End, which is the globally optimal route with a cost of 8 per agent. Therefore, larger random deviation to other routes will only increase the average cost. When 40 agents are involved, the Nash equilibrium is that everyone stays on Start - A - B - End, which is the worst case possible with a cost of 80 per agent. Therefore, larger random deviation from the Nash equilibrium will decrease the average cost.

3.4 Super Fast Highway vs. No Highway in Mixed-Agent Settings

Each run only involves one type of agent, and 36 agents are used per run. Cost is averaged over 100 turns and over every agent type.



With different types of agents, the Fictitious-Play agents seem to be able to harvest the randomness generated by the other two types of agents. In this case, the Fictitious-Play agents have the smaller cost after adding the super highway.

REFERENCES

[1] https://www.cs.bham.ac.uk/internal/courses/robotics/lectures/ucb1.pdf
[2] https://en.wikipedia.org/wiki/Braess%27_paradox