Reinforcement Learning Applied in a Police Pursuit

Bernardo Eichler (77988)

Ricardo Sequeira (79750)

1. INTRODUCTION

Our project consists in four police cars pursuing a thief car in a particular city. The objective is to catch the thief, traping him between two cells from which he can't run away. The police cars respond to a police station that controls all the pursuit by registering the thief position when he is seen by one of the agents (since his position is unknown in the beginning of the problem).

2. AGENTS : POLICE CARS

Our main agents are the police cars because they are the entity to be studied in this problem. Our goal was to understand what is the best solution to catch a wanted car (in this case a thief car): should the police car go after the thief since the moment he sees it? Or should he do some barricade in the end of a street to trap him?

2.1 Sensors

A police car knows his position and where can he head to (his possible directions). He even recognizes the city boundaries from a distance of 3 cells. The same distance from which he can identify the thief. The rest of the information that he gets is from the police station that he is connected and which informs him if the thief position is known or not, and if it is, where is him.

2.2 Actuators

The police car can move around the map in any legal direction (legal directions are embedded in the map cells) to another cell that isn't occupied by another car. There is this special case when the police gets to know a thief position (through the police station communication) and can choose a illegal direction (a direction that isn't embedded in the cell but allows him to get to another road cell inside of the map).

3. ENVIRONMENT: CITY AND THIEF

We wanted to have a rich environment so we envisioned a city with civil cars driving to given legal directions (embedded in map cells) leaving and entering again in the map (normal traffic). However the difficulty of our problem given those conditions was too high so we cut our vision for the city and we ended with a close simple map (police and thief don't leave the city because the city is closed for the pursuit) without civil cars (but the source code for them is there).

The thief is implemented as a car just like the police ones, with similar sensors and actuators. That said, we can say that the thief movement is completely random and only changed by the position of other cars (in this case, there are only other police cars). This last information will be important for our analysis.

The thief was intended to be searching for a garage which would grant him a way out of the city and be succeed in the pursuit. But once more, this proved to be a big complexity in our problem so we cut it and made the thief position in the map random as already said. The thief has also the capacity of choosing (randomly) illegal directions when trapped by the police.

4. REINFORCEMENT LEARNING

Since the beginning that our intention was to use reinforcement learning in the decision making algorithm, in particular, using the Q learning technique to calculate the quality of every police car move. The common rewards for every agent is assured by the police station which distributes the reward value.

4.1 Reward Function

The reward function that we implemented gives a maximum of 1000 if the police car has directly arrested the thief, gives 200 if the police car is moving to a cell in the thief direction (if his position is known) and finally, give a bonus reward for how recently the cell of his next position was visited: if the cell was recently visited the reward is minimal here, if it was never visited (or in the last 10 steps) the reward is maximum.

4.2 Adjustements

The direction decision during the train iterations is not completely random because we made the police cars to choose the best action (calculated until there) half of the times and choose a random action the other half (to not get stuck in local maximum).

4.3 Problems

After the training with every number of iterations (we tried with lots of them so that was not the problem) we concluded that we have a problem with our Q. It's possible that is due to our reward function being too simple or maybe because the thief position is completely random according to the position of the police cars. We tried to fix our Q but we failed in the end and we couldn't pass to the next step.

5. THE NEXT STEP

We wanted to, after fixing the decision making algorithm, implement the police station as a unique agent who would decide to call more police cars (at a financial cost) given the time that a given pursuit is taking. With that, we wanted to study the problem of how many police cars are needed and the perfect time for a pursuit. We wanted also to give the police cars a personality: a more selfish one (who wants to be rewarded for catching the thief) or a more collaborative one (who wants to help catch the thief and reward everyone).

6. CONCLUSIONS (UPDATED)

This week meeting with Professor Rui Henriques was very productive because he pointed the fact that our vision of a State in the Q learning implementation didn't include the thief position. This means that the agents (the police cars) weren't making decisions taking in account the position of the thief in the map.

We were confident that this was one of the main problems with our reinforcement learning so we changed the concept of state (it was only the position of the police and only that) to a new MapState with not only the position of the police but also the thief one. However there are a big number of possible combinations if we consider all map points for the two cars (more precisely: 32x32x32x32 combinations). To reduce the list of states, we saved a list of only road cells of the map (a total of approximately 450 cells) and only used those for the possible combinations of police and thief positions.

With this MapState, we finally saw some results after the Q learning that allow us to make one little analysis about the time spent in the thief pursuit (if even successful in a certain limit of time that we established as 100000 microseconds) and the number of police cars involved/needed. All this was intended to be the Police Station objective: manage the thief pursuit in terms of time and money (for police cars reinforcements) spent.

To have something to compare with our Q Learning results, we returned to our expected best policy which was for the police cars to choose always the direction of the thief, no matter what.

6.1 Policy Always Chase Thief vs Q Learning Technique

All the data was obtained through 1000 iterations of the problem with the police cars and the thief starting always in the same positions (with a random movement afterwards). We considered a failed pursuit if the total time was greater than 100.000 microseconds.

6.2 Discussion of Results

What can we conclude from the results? Obviously, the policy of always chasing the thief is not a bad policy, in fact it probably is one of the best policies (except if we have 2 police cars because in that case, we have a failed percentage of 70%).

Probably almost everyone would say that the results prove that the Policy Always Chase Thief is better than the Q

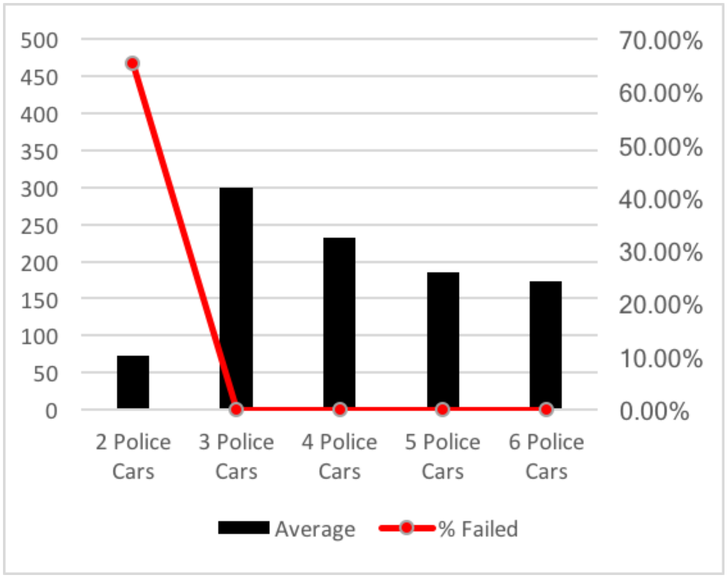

Figure 1: Results of the Policy Always Chase Thief

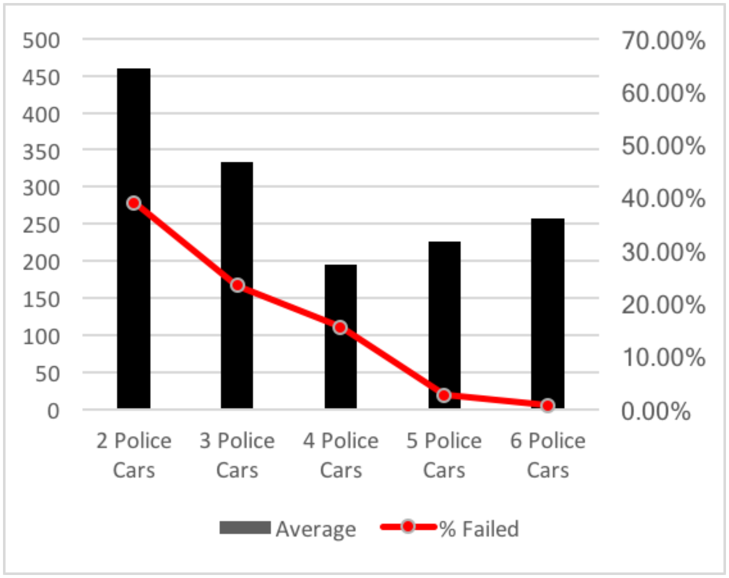

Figure 2: Results of the Q Learning Technique

Learning Technique. However, we think that the results are more interesting than that because if we don't considered the failed iterations, the Q Learning Technique wins in time and cars spent on the pursuit (with the 4 police cars example).

What we mean is that our Q Learning must have some problematic specific situation in the map that is causing a strange high failed percentage. We didn't have time to explore what could it be but is correct to presume that if that error was corrected, we would get better results than the Policy Always Chase Thief.

6.3 Some Other Ideas

We really wanted to include the civil cars in the map because we think that this addition would highlight the Q Learning advantages (in comparison with the Policy Always Chase Thief). While trying to improve our project we thought about making a MapState where the police car knows what cells of the map are blocked by civil and other police cars (a simple boolean for every road cell). Unfortunately, there was no time for that but we are certain that would make the previous analysis and comparison much more interesting.