COURSEWORK REPORT:DYNAMIC MAZE SOLVING

Abhilash Pulickal Scaria
aps1n21@soton.ac.uk

ABSTRACT

This work involves using a reinforcement technique to solve a 199x199 dynamic maze and observing the results. An Epsilon Greedy Q Learning algorithm was used to solve the maze. The approach taken and the parameters used are described. The result observed is also analysed.

1 PROBLEM STATEMENT

Using reinforcement learning compute the minimum time path from position (1,1) to (199,199) for a 199x199 maze with dynamic components. The environment described has static walls and dynamic fire element. The agent can observe eight of its surrounding tiles and its own location at a time using a local information function. The local information function gives locations of walls and it also gives the location of fire and its timing information(how long the fire lasts). The agent can move up, down, left or right, it can also choose to stay at a location.The agent should move from (1,1) to (199,199) avoiding the walls and fire.

2 METHODOLOGY AND RESULTS

The reinforcement learning algorithm chosen to solve the static part of this problem is Q learning(Watkins, 1989). A Q table with q values for each state action pair was constructed. The best action to take at each state is given by this Q table. The Q values are updated using the Bellman equation.

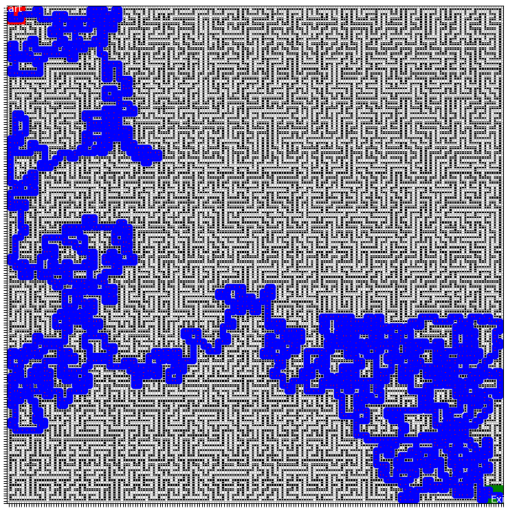

Figure 1: The minimum time path from (1,1) to (199,199) traversed by agent

An epsilon greedy approach was used to select the action to perform. The action to take is selected using this epsilon(exploration rate). If the exploration rate is more than a random value, a random

action is chosen else the action to take at that state predicted using q table is chosen. This approach allowed for a good exploration-exploitation trade-off in this problem. Initially the exploration rate was set to 0.1 which didn't allow the agent to explore much and caused the algorithm to take too much time to converge. After some trial, the exploration rate was set to 0.2. An exploration decay was also applied to reduce the exploration rate after each episode. This was to ensure the agent doesn't wander off in the later episodes. The exploration decay was set to 0.995, after each episode the exploration rate becomes 0.995 of its earlier value. A reward of 10 was given for reaching goal. Penalty for moving to walls and previously visited cell was given as -0.75 and -0.25 respectively. A penalty of -0.05 was given for each move done. The convergence was checked using the winning rate. Once the model starts winning continuously in each episodes it was inferred that it has reached a convergence.The model was run for different number of episode and it was observed to converge around 300 episodes.The model took around 18-20 minutes to converge. The minimum time path obtained is given in figure 1. The code can be run by running the main.py file

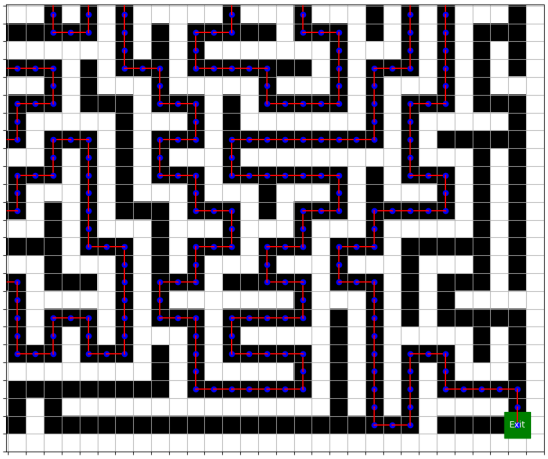

Figure 2: Zoomed view of area around the goal

A Deep Q Network model with 2 hidden layers with 64 hidden units was also constructed to solve this problem, but even for smaller 8x8 maze the model took 4+ hours to converge. For the 199x199 maze each episode was taking over 10 hours. Due to the enormous amount of time required for this model to converge it was not further pursued.

3 Conclusion

An epsilon greedy Q learning was implemented to solve the 199x199 static maze problem. The parameters of the model were tuned and the model was found to converge in a respectable time. A minimum time path was computed and the result was plotted.

References

C. J. C. H. Watkins. *Learning from Delayed Rewards*. PhD thesis, King's College, Oxford, 1989.