RUTGERS UNIVERSITY, NEW BRUNSWICK

Mazes

Eshaan Gandhi, Siddarth Mandayam

September 30, 2020

1 THE MAZE IS ON FIRE

1.1 Testing out our environment

What I mean by testing out our environment is that we wanted to demonstrate the maze being generated with a given block density ρ. We also wanted to demonstrate the placement of fire and initial path to the target.

So we first generated a few maps with varying levels of block-density as demonstrated by the graphics below.

Next we would like to confirm that one element at random gets set on fire as demonstrated by the graphics below.

1.2 Strategy 1

So strategy 1 is a naive way to go about this, but would not require that many resource to go about it. It simply does not care about the fire, and the person does get burned pretty often. Here is the strategy statement in detail. At the start of the maze, wherever the fire is, we solve for the shortest path from upper left to lower right, and follow it until the agent exits the maze or burns. This strategy does not modify its initial path as the fire changes.

We were tasked in generating a plot of 'average successes vs flammability q'. To do that we had to run our program for strategy 1 n number of times. In the assignment document, the minimum number is 10, but we realized that these many tries does not represent the true statistic well because of variance. So we in turn ran the maze a 1000 times for 11 values of q - 0 .. 1.

Table 1.1: Strategy 1

Valid iterations	q	s - number of succeesses	u - number of times the agent got burned
1000	0.0	736	264
1000	0.1	556	444
1000	0.2	375	625
1000	0.3	316	684
1000	0.4	181	819
1000	0.5	162	838
1000	0.6	95	905
1000	0.7	75	925
1000	0.8	59	941
1000	0.9	42	958
1000	1.0	37	963

We have recorded our finding in the table above.

We then plot our findings in a line graph below.

1.3 Strategy 2

So strategy 2 is a better way to go about this than strategy 1, but would require more resource than strategy 1. It recomputes it's path every time it makes a step. Here is the strategy statement in detail. At every time step, we re-compute the shortest path from the agent's current position to the goal position, based on the current state of the maze and the fire. We then follow this new path one time step, then re-compute. This strategy constantly re-adjusts its plan based on the evolution of the fire. If the agent gets trapped with no path to the goal, it dies.

We were tasked in generating a plot of 'average successes vs flammability q'. To do that we had to run our program for strategy 2 n number of times. In the assignment document, the minimum number is 10, but we realized that these many tries does not represent the true statistic well because of variance. So we in turn ran the maze a 1000 times for 11 values of q - 0 .. 1.

Valid iterations	q	s - number of succeesses	u - number of times the agent got burned
1000	0.0	913	87
1000	0.1	706	294
1000	0.2	451	549
1000	0.3	353	647
1000	0.4	241	759
1000	0.5	172	828
1000	0.6	134	886
1000	0.7	70	930
1000	0.8	68	932
1000	0.9	46	954
1000	1.0	52	948

Valid iterations	q - flammability rate	s - succeesses	u - Times the agent got burned
1000	0.0	919	81
1000	0.1	651	349
1000	0.2	453	547
1000	0.3	337	663
1000	0.4	214	786
1000	0.5	159	841
1000	0.6	99	901
1000	0.7	74	926
1000	0.8	54	946
1000	0.9	58	942
1000	1.0	39	961

We have recorded our finding in the table above.

We then plot our findings in a line graph below.

1.4 Strategy 3 - GTFO! (Graph The Fire Out)

We then wanted to try out our own strategy. The problem with strategy 1 and strategy 2 is that it does not take into consideration changing state of the maze and future state of the maze respectively. So we thought we could simulate part of the maze and see how we do. We started out simulating the whole maze but that would just mean that the fire engulfs the whole maze and finding a path is near impossible. So we simulate a part of the maze and try to find a path. We then take that path in the original maze and would likely make it. If we do make it, then we simulate it again. This strategy takes the future states into account, moves a bit, and then takes into account the changing state of the fire.

Well we found out it does very well for when the fire is not spreading rapidly, but does rather poorly when the fire is pretty fast. Here is the data.

1.5 Comparison

If we look at all the data, we can conclude that strategy 2 might be the best strategy, but it is very computationally expensive. We calculate the BFS path at every step. Our player might not have that kind of time when the fire is spreading fast.

For the first strategy the time complexity is $O(b^{d+1})$ as we only compute BFS once. Where $b = 4$ as we are computing 4 adjacent nodes.

Now if we look at Strategy 2, we are doing BFS on every movement. That is $O(k(b^{d+1})$ In real time we probably don't have that much time to make a decision in a maze that may be burning pretty fast.

The third strategy is simulating the future fire. If we were in a maze, we could kind of eyeball what the fire would look like. Simulating fire is not that computationally heavy and we only have to do BFS a fraction of the time we do it in strategy 2. We also get very similar results if we go about strategy 3.

1.6 Note on dimensions

As we mentioned the time complexities of all three methods, this becomes very computationally expensive for the computer. I also wanted to generate a lot more maps than 10 as that removes variance. There are a number of times when we generate a map and it is rejected as there is not even an initial way to the goal. For these reasons, I chose a non-trivial dimension that is fast enough to run more than 1000 times for each value of q and also large enough for the variance to not produce too much change how many successes we get.