CS 520: Exploring Search

16:198:520

1 This Maze is on *Fire*

A **maze** will be a square grid of cells / locations, where each cell is either empty or occupied. An agent wishes to travel from the upper left corner to the lower right corner, along the shortest path possible. The agent can only move from empty cells to neighboring empty cells in the up/down direction, or left/right - each cell has potentially four neighbors.

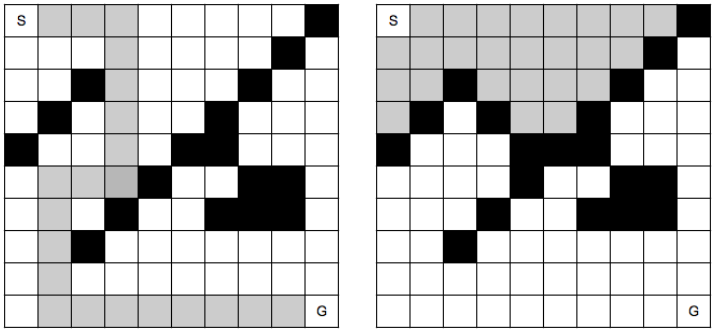

Figure 1: Successful and Unsuccessful Maze Environments.

Mazes may be generated in the following way: for a given dimension **dim** construct a **dim x dim** array; given a probability p of a cell being occupied ($0 < p < 1$), read through each cell in the array and determine at random if it should be filled or empty. When filling cells, exclude the upper left and lower right corners (the start and goal, respectively). It is convenient to define a function to generate these maps for a given **dim** and p.

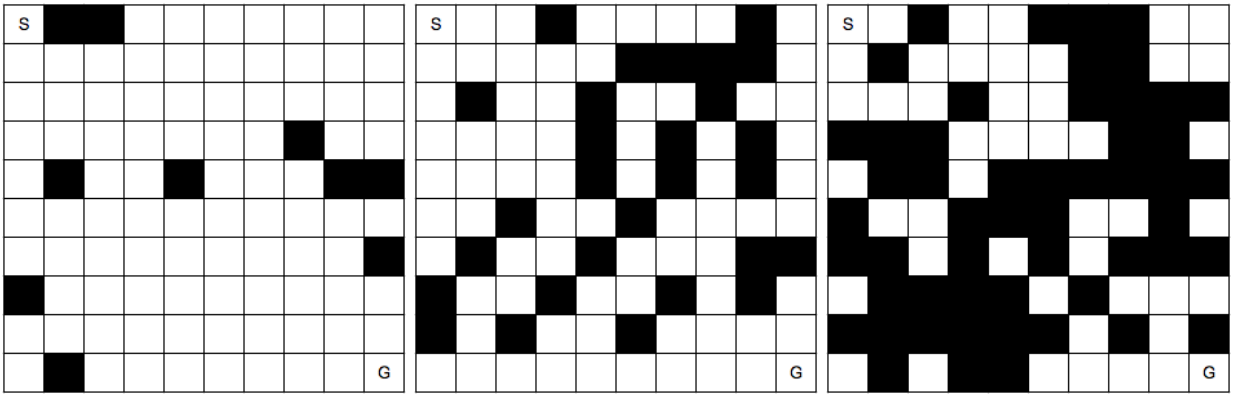

Figure 2: Maps generated with $p = 0.1, 0.3, 0.5$ respectively.

However, we are are not just interested in solving static mazes - these mazes are on *fire*, actively burning down around you. We model this in the following way: for every move the agent makes, the fire then advances (described below), and they alternate back and forth. The agent would like to exit the maze prior to running into the fire. Under this model, while an agent might utilize a search algorithm to generate an optimal path through the the *current* state of the maze (analyzing the current state and planning a future path through it), it will not necessarily be valid for *future* states of the maze. This means that while the agent may plan a short path through the maze, whether or not it will be able to take that path depends on the fire. In this case, the environment is dynamic, changing over time.

We model the advancement of the fire in the following way: any cell in the maze is either 'open', 'blocked', or 'on fire'. Starting out, a randomly selected open cell is 'on fire'. The agent can move between open cells or choose to stay in place, once per time step. The agent cannot move into cells that are on fire, and if the agent's cell catches on fire it dies. But each time-step the agent moves move, the fire may spread, according to the following rules: For some 'flammability rate' $0 \leq q \leq 1$

- If a free cell has no burning neighbors, it will still be free in the next time step.
- If a cell is on fire, it will still be on fire in the next time step.
- If a free cell has k burning neighbors, it will be on fire in the next time step with probability $1 - (1 - q)^k$.

Note, for $q = 0$, the fire is effectively frozen in place, and for $q = 1$, the fire spreads quite rapidly. **Additionally, blocked cells do not burn, and may serve as a barrier between the agent and the fire.**

How can you solve the problem (to move the agent from upper left to lower right, as before) in this situation?

Consider the following base line strategies:

Strategy 1) At the start of the maze, wherever the fire is, 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.

Strategy 2) At every time step, 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. 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.

For each strategy, for a given dimension (*discussed below*) and obstacle density $p = 0.3$, generate a plot of 'average successes vs flammability q'. Which is the superior strategy? For each test value of q, generate and solve at least 10 mazes, restarting each 10 times with new initial fire locations. **Note:** Please discard any maze where there is no path from the start to the goal node. Additionally, please discard any maze where there is no path from the initial position of the agent to the initial position of the fire - for these mazes, the fire will never catch the agent and the agent is not in danger.

Come up with your own strategy to solve this problem (+5 points are available if you come up with a clever acronym for your strategy), and try to beat both the above strategies. How can you formulate the problem in an approachable way? How can you apply the algorithms discussed? Note, Strategy 1 does not account for the changing state of the fire, but Strategy 2 does. But Strategy 2 does not account for how the fire is going to look in the future. How could you include that?

A full credit solution must take into account not only the current state of the fire but potential future states of the fire, and compare the strategy to Strategy 1 and Strategy 2 on a similar average successes vs flammability graph. Additionally, while your strategy may increase survivability, analyze the time costs of your strategy compared to the other two - in a real burning maze, you have limited time to make your decisions about where to move next. Be clear and explicit in your writeup, presenting your reasoning and ideas behind your strategy.

A Note On Dimension: Note at least three bottlenecks that may arise in generating data: a) making sure the current maze is solvable, b) making sure there is a path from the agent to the initial fire source, and c) whatever shortest path computation is being done in Strategies 1, 2, and your strategy. Note bottleneck a) and b) do not require the shortest path, and bottleneck c) should be handled in such a way as to minimize the amount of work necessary. How can these be handled algorithmically, to let you work with the largest possible mazes you can? This should factor into the analysis and writeup of your work.

2 Maze Thinning

In this section, the maze is no longer on fire.

As discussed in class, the A^* algorithm minimizes the number of nodes in the search space that need to be explored (subject to the quality of the heuristic). Recall the idea of the heuristic is to estimate (ideally, from below) the remaining cost from the current location in search space to the target. In class, we discussed several means for generating heuristics; in particular, we focused on geometric or geographical heuristics (euclidean distance, Manhattan distance), and heuristics based on relaxations (solving simpler versions of the problem). Geometric heuristics are easy to apply to the maze problem, but can we apply relaxations as well?

Consider the following relaxation: for a given maze, remove some fraction ρ of the maze's obstacles at random. Any path through the original maze will be valid in this 'thinned' maze, but there will be more feasible paths through the thinned maze and therefore it is 'easier' to solve, potentially using something like A^*-Manhattan. In particular, the distance from any point in the thinned maze to the goal is a lower bound on the distance from that point to the goal in the original maze. As such, shortest path lengths in the thinned maze can be used as a heuristic in the original maze.

- **Potential Advantages:** Because the solution to the thinned maze is based on actual maze obstacles, the length of it will more accurately estimate the remaining distance than something like the Manhattan distance, giving us tighter lower bounds and more direction for our bootstrapped A^*.
- **Potential Disadvantages:** Because the value of the heuristic is based on solving another maze, computing the heuristic potentially becomes computationally expensive. The maze would have to be thinned enough that the thinned version is easy and efficient to solve, while not being too thinned it doesn't provide useful information.

Try to determine whether or not this is a viable strategy. For an obstacle density of $p = 0.3$, write an A^*-Manhattan solver and determine the largest size maze you can reliably, repeatedly solve in a reasonable amount of time. Then implement the ρ-thinning strategy as an alternative solver, bootstrapped off the first.

Graph, as a function of ρ, the fraction of nodes in the primary maze (non-thinned) expanded when solved by this thinning strategy (for each test value ρ, generate and solve at least 100 mazes). Are there any ρ where you see a savings over the straight A^*-Manhattan at this obstacle density? What additional time costs do you incur? Is the strategy viable?

Try to construct and analyze another relaxation-based strategy for generating a heuristic. Do you think this approach is viable? Why or why not. *How does the representation of the maze factor in to this (sparse vs not sparse, etc)? Because certain strategies might have favorable or unfavorable regimes, could the strategy be variable based on how close you are to the goal (via Manhattan distance) or something similarly dynamic? How could the remainder of the maze be 'approximated' in some way to produce a useful solution?*

Be thorough, be clear.