Victoria Li - vl4kz
Maurice Wong - mzw7af

Pathfinding: An Exploration of A* and Uncertainty

1. Basic pathfinding algorithm

Our basic pathfinding algorithm was the A* search algorithm utilizing diagonal distance as a heuristic function. In our algorithm, we began by traversing all possible partial paths from the starting point and then calculating the cost function $f(n) = g(n) + h(n)$ for each of those paths. In each case, our *g(n)* was the best possible cost to travel to point *n* found so far. Our *h(n)* was calculated as followed:

```
public double diagDistance(Point p1, Point p2) {
      double dx = Math.abs(p1.getX() - p2.getX());
      double dy = Math.abs(p1.getY() - p2.getY());

      return ADJ_COST * (dx + dy) + (DIAG_COST - 2.0 * ADJ_COST) * Math.min(dx, dy);
}
```

ADJ_COST represents the cost of moving to a cell to the north, south, east, or west of your current cell. DIAG_COST represents the cost of moving diagonally.[1]

For each iteration of the algorithm, the cell with the lowest *f(n)* value is considered, and then all of its neighboring cells' values of *f(n)*, *g(n)*, and *h(n)* are updated accordingly. This continues until all cells are popped off the queue or the goal cell is found. Then a path is reconstructed from a mapping of nodes to their parent nodes (the nodes that they came from), and the robot is moved down the final path found.

1.1 Analysis

For the provided data sets MyInputFile 1-4, the most efficient path was found with the following performances:

1. 4 moves, 74 pings
2. 8 moves, 228 pings
3. 14 moves, 876 pings
4. 39 moves, 1709 pings

From these data sets and their sizes, bigger grids require broader searching for the A* search algorithm, resulting in more pings. Additionally, it was found that more "false paths" led to more pings, even for data sets of the same size. If there were paths that seemingly led to the end position but failed to do so in the end, more pings were required. This is because the algorithm continually tries to expand along a possible best branch before pursuing others, so it attempts to search these misleading paths before determining it is inviable. This can be demonstrated in myInputFile7, which is similar to myInputFile4 with the exception that there is one longer "false path" that runs around the perimeter of the grid. This grid required 2407 pings

[1] http://theory.stanford.edu/~amitp/GameProgramming/Heuristics.html

compared with the original 1709 pings, showing that the presence of false paths lead to more pings and inefficiency.

2. Adapting to uncertainty

To deal with uncertain situations, we attempted to shorten the path found by our A* search algorithm. Instead of pinging the entire map, finding the most optimum path, and then finally moving the robot, we opted to ping the map to find the best path up to a certain distance away (e.g. six cells) and then move the robot down the partial path found, marking the tiles that it had already traversed to avoid repeated searching over spaces.

Since we were not sure if pings were accurate, we could not fully ignore paths that were seemingly blocked by X's since they might not be accurate. In the normal A* algorithm for the certain case, we ignore the X's completely since they are known to be blocked. Because of this uncertainty, we instead assigned a high cost value *f(n)* to spaces that were pinged as X's. This would result in considering these spaces last in our algorithm.

Additionally, partial paths that were found were not always possible because points in the path that were pinged to be O's could in reality be X's that were blocked. Our algorithm adapted to this by moving the robot as far down the partial path as possible. If it was found that the path was actually blocked, then it would restart the partial path A* algorithm from its current location and attempt to find another partial path.

In the event that the robot ended up in a dead end where all of the neighboring tiles were either X's or spaces already traveled, the robot would backtrack along the spaces it had traversed until it was at a space that did not have blocked neighbors, and the A* algorithm would restart from there to explore the unblocked paths. This mimics depth first search, guaranteeing that the robot will be able to recover from misleading branches and eventually search all possible paths until it finds the end position.

To further optimize our algorithm, we also tried to prune out paths/branches that were not possible. We kept a set of points that had already been attempted by the robot that were known to lead to dead ends/blocked paths. Using this knowledge, our A* local search algorithm was able to prune branches that were known to lead to blocked paths, reducing the number of moves necessary.

3. Data from different inputs

Running our uncertainty algorithm on the 4 given test cases, these were the results in one run, although the results are not deterministic because of the probabilistic factor:

1. 4 moves, 74 pings

Victoria Li - vl4kz
Maurice Wong - mzw7af

Pathfinding: An Exploration of A* and Uncertainty

2. 10 moves, 396 pings
3. 40 moves, 3001 pings
4. 110 moves, 7671 pings

We also made our own test cases: myInputFile5, myInputFile6, and myInputFile7.

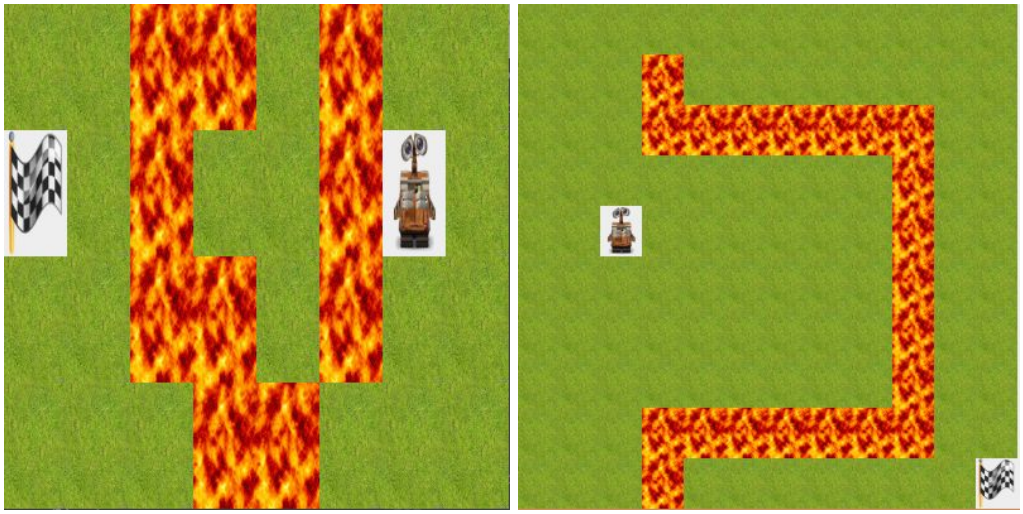

Figure 1. myInputFile5 and myInputFile6

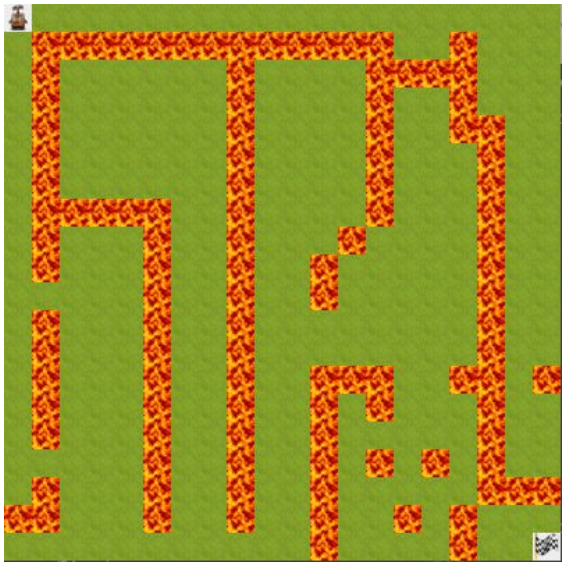

Figure 2. myInputFile7

In the case of myInputFile5, our uncertain algorithm exits out once it determines that there is no possible path that will lead to the goal. For myInputFile 6, which is test case 3 modified to have only one possible opening, the results were 70 moves, 3802 pings. When most of the possible paths to take are blocked off by obstacles, efficiency of the algorithm decreases as many of the robot's tried partial paths are blocked once the robot reaches the end, and the robot must backtrack until it can find the path that leads to the one opening.

Victoria Li - vl4kz
Maurice Wong - mzw7af

Pathfinding: An Exploration of A* and Uncertainty

Finally for myInputFile7, which is test case 4 modified to have a longer misleading path, the results were 241 moves, 13319 pings. When misleading paths are present, the efficiency of the algorithm significantly worsens since the robot must physically traverse down these false paths and backtrack once it discovers they are false.

3.1 Changes made to the algorithm

The most influential parameter that can be tweaked in the uncertain algorithm is the number of spaces away from the current point that our partial path A* algorithm searches before allowing the robot to move to the best space found. We found that adjusting this parameter had trade-offs between the number of moves and pings made. When this parameter is higher, the number of moves made decreases because it is able to search further away decide on a better path to follow. However, it also means that there is higher uncertainty with tiles it checks further away and there are more possible branches to consider before moving, resulting in an increased number of pings.

The opposite is true when the parameter is decreased. The number of moves increases because the robot cannot check longer paths to determine their success and resultingly makes more incorrect moves, but the number of pings decreases since the A* algorithm doesn't have as many branches to search in a decreased search radius.