Caroline Holmes and Eamon Collins
Diochnos
AI HW 2 write-up

Our pathfinding algorithm under certainty is the A* algorithm, with our heuristic being the length of the minimal possible path to the goal given no impediments. We have figured this to be max(dx, dy), where dx and dy are the difference between the x and y values of the current position and the goal node. A* is a variation of Dijkstra's, so it guaranteed to find the shortest path given that the heuristic is consistent, and attached to this writeup is our proof that our chosen heuristic is indeed consistent. It can achieve better performance than Dijkstra's, as the heuristic allows it to prioritize paths that are more likely to be the shortest.

The more the path to the goal node resembles the best possible path (that is, a path with no obstacles between the start and goal points) the more efficiently our algorithm will calculate the path. This is because we are using the best possible number of moves given no impediments as our heuristic, so in a case where there are no obstacles the f-scores along the path will not increase, as every movement will be matched in a reduction in the heuristic. Thus it will not consider any avenues off of one of the optimal paths and will effectively go straight to the goal node.

To deal with uncertainty, we ping all the spots on the board a certain initial number of times, and then from there we look at which squares we are not certain enough about. We define a decision threshold in terms of standard deviations of the distribution at that point. If the tile does not fall within the threshold for either X or O, we ping-once-test-once until the results fall within our standards for a decision. Moving this decision threshold down (and therefore narrowing the acceptable range for a decision) increases how confident we are when making a decision about each tile, at the tradeoff of increasing the number of pings it is likely to take to get to that level of certainty. We have found through modulating both the initial number of pings before testing and the stringency of the test that what we need is a threshold of around 1.4 standard deviations and a number of initial tests just big enough to make that meaningful in most cases, we use around 150-170 pings per tile and get about 5000 pings on top of that in dynamically added pings for tiles that didn't meet our standard within those pings. We like this because on top of the 60,000 or so pings we're getting already, a few more to be certain doesn't seem so bad, especially when we almost always achieve optimality on the largest test input we were given. In the case that something was actually evaluated incorrectly along the path that prevents the path from working, the robot will redo the whole pinging and testing process from the current position, doubling the number of pings necessary to complete the path but usually only adding 2 or so moves to the total count.

Some example runs of our uncertain algorithm on inputFile4 with varied parameters:

Initial Tests	Decision Threshold	Number of Moves	Number of Pings
160	1.4	38	67554
160	.5	38	184294
160	2.5	38	128010

As can be seen in the table, too narrow of a decision range and the pings spike due to the amount of certainty required to make a decision about any give tile being too high. Too wide of a decision range and the pings can go up due to a loss of certainty, resulting in an error that requires recalculation of the path, each recalculation requiring a large number of pings. It is worth noting that an error in the path often results in a suboptimal final path, but not necessarily.