**Data and Analysis for Project 1**

**Bot 4 Design**

My Bot 4’s algorithm is somewhat of the previous bot’s strategies combined but instead of breath-first search, I’m using A\*, which seems to make the algorithm extremely more efficient.

Using the knowledge it has from Bot 2 and 3 to avoid the fire as well as some of its adjacent blocks and re-pathfinding whenever a fire is blocking the path, the bot is able to get to the button with quite a high success rate. Additionally, the heuristic I decided to use for bot 4 is the Manhattan Distance between two coordinates in the grid, so now it uses both distance from the start and the estimated remaining distance to find the shortest path to the button, unlike the previous bots which are just blindly searching the entire grid.

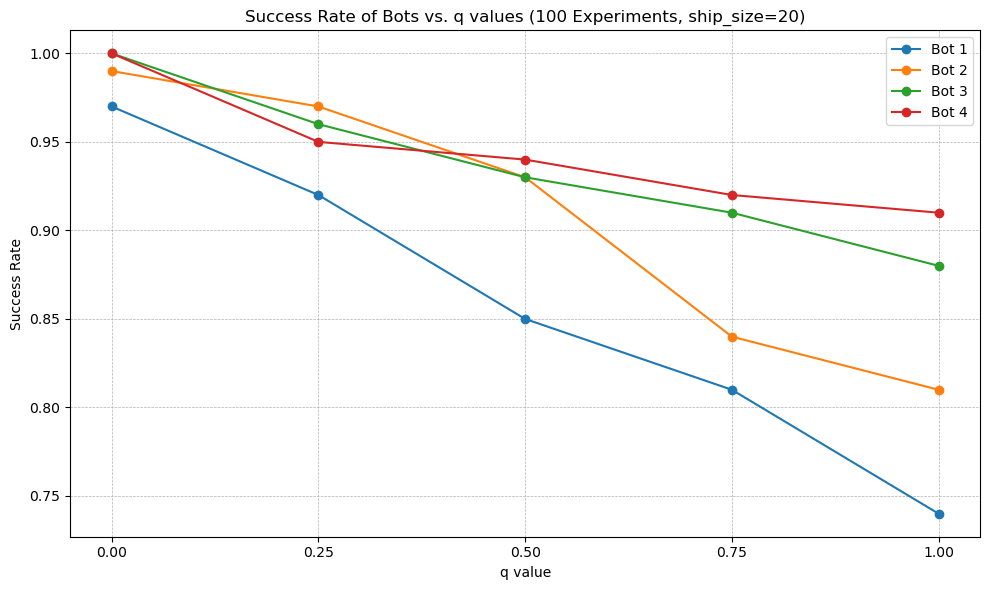
**Testing Results**

To test the performance of my bots, I ran several simulations with different paramters to collect data on the success rates.

For each simulation, I chose to generate a 20x20 ship since larger ones might take a lot longer. I only ran 100 simulations for each q on each bot due to time issues, in the future I definitely plan to run more to make my data more accurate. Speific data points can be found in the testing\_results jupyter notebook.

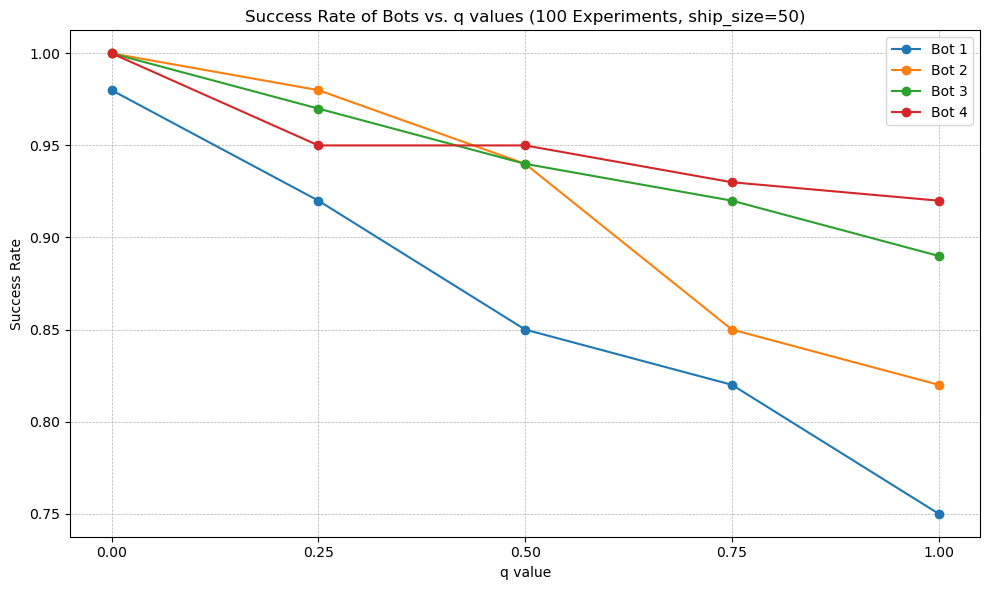
The different q values I decided to use are: q: [0, 0.25, 0.5, 0.75, 1]

After running all these simulations, using matplotlib, I graphed all the data points. As we can see from the graph below, as the q values increases the success rates drops. This is expected since higher the q value, faster the fire spread, meaning less chances for the bots to get to the button before the fire.



Next, I also wanted to see how they would perform on larger grids. I chose to use 50x50 grids, since this takes quite long to generate, I pre-generated and stored 100 50x50 grids in a folder and used them for the simulations below.

For each bot, I used the same 100 grids but different initial button, bot and fire locations to see the performance. As we can see since the grid is larger, there is better chances for the fire to spawn away from the button and for the bot to navigate to the button before the fire, making the success rates larger.



**Conclusion**

Most of the times, when the bots fail, either the fire gets to the button first, or the fire completely surrounds the button, causing all paths from the bot to the button to be blocked. There are certainly better decisions in some situations they could’ve made, for instance, if they had chosen a better path at an earlier stage, they could’ve gotten to the button before the fire spread there. But in other situations, for instance if the fire spawns a lot closer to the button than the bot, there is nothing the bot can do to reach the button before it’s destroyed.

**Planning the Ideal Bot**

One thing that we could possibly do to increase the success rate even more is to predict the direction or the precise location of the next cell that is going to catch on fire, since from the equation the more burning neighbors that a cell has, the more likely its going to catch on fire. Bot 3 and 4 only tries to avoid the fire by not going anywhere near the cells that are burning which could make the path longer. For instance, if the fire somehow spreads the same or similar direction as the bot moves, the bot might end up wrapping around the fire since it doesn’t know what the fire might do next, but if at an earlier timestep the bot has taken a path that’s opposite to the fire spread, it certainly would be shorter.

Another different approach is optimizations for the A\*/BFS searching algorithm for this specific task, for instance the heuristic that I’m using is the Manhattan distance, maybe there is something else that is better such as the Euclidean distance.