CS520: INTRODUCTION TO AI

Project 1 Report

The Bot is on Fire

By,

Chiranjeevi Bhuvaneshwari Pamarthy [cp1270]

Shweatha Bathina Mallikarjuna [sb2808]

Table of Contents

1.Abstract

This project examines different strategies for a bot navigating a fire-prone spaceship to reach a suppression button. Since fire spreads unpredictably based on its surroundings, the bot must constantly adapt to survive. The study evaluates four approaches: a **static path bot** that follows a precomputed route, a **reactive bot** that recalculates paths in real time, a **cautious bot** that avoids high-risk areas, and an **optimized bot** using **Dijkstra's algorithm with fire risk prediction** to make smarter decisions.

Each strategy is tested across different fire spread conditions to measure adaptability and success rates. The results show that **bots with real-time planning and fire prediction significantly outperform static approaches**. This study highlights the importance of **dynamic pathfinding, risk assessment, and adaptability** in uncertain environments, offering insights relevant to robotics, AI navigation, and emergency response systems.

2. Introduction

2.1 Ship Generation – The Environment

To set up the spaceship environment, we generate a **40x40 grid** with a mix of **walls and open paths**. The goal is to create a unique and solvable maze where the bot must navigate to reach the fire suppression button while avoiding the spreading fire.

How the Ship is Created

1. **Creating Paths and Walls**
    a. The grid starts with a random **open cell**, which expands outward to form walkable paths.
    b. Walls are placed to shape the maze, ensuring there are **obstacles but still a way through**.
2. **Fixing Dead Ends**
    a. Some paths may lead to **dead ends**, making movement difficult.
    b. About **half of these dead ends are opened** to ensure there are always multiple possible routes.
3. **Placing Fire, Bot, and Button**
    a. Three key elements are placed **randomly in open areas**:
        i. **Fire Source** – Where the fire starts.
        ii. **Bot Position** – Where the bot begins.

        iii. **Button Location** – The target that stops the fire.

    b. These are placed in different spots to make each scenario unique.

4. **Saving and Using the Layouts**
    a. Each layout is checked to make sure the bot **can reach the button**. If not, it is discarded.
    b. A total of **1,000 different ship layouts** are generated and stored in a **CSV file** for use in simulations.
    c. The grid consists of **walls (0), open paths (1), fire sources (2), the bot (3), and the fire suppression button (4)**, creating a dynamic and engaging test environment for different bot strategies.

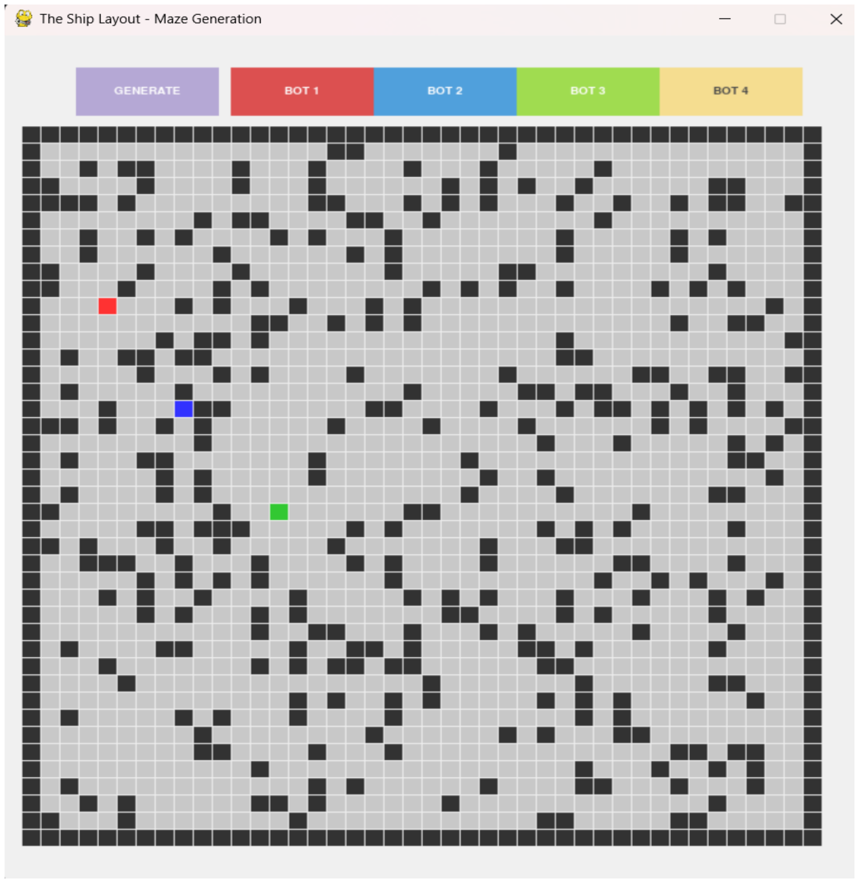


Fig.1 The ship Layout 40 x 40

2.2 The Fire

In the ship environment, fire starts at an open cell picked at random and progressively extends to surrounding open areas. The flammability factor (q) and the quantity of burning neighbors a cell has affect the rate of spread of it. Depending on the count of surrounding burning cells, any non-burning open cell has a chance to catch fire at every time step. More nearby cells already on fire raises the likelihood of ignition. This method runs concurrently across the grid to guarantee that fire spreads realistically free from impact of past modifications inside the same phase. Conversely, a reduced flammability factor slows down the spread and provides more navigation time. Any bot negotiating the ship will find this fire simulation to be a major obstacle since it adds an element of uncertainty and requires a safe path while avoiding the expanding flames.

2.3 Bots

BOT 1

Bot 1 is a straightforward strategy by precomputing the shortest path to the fire suppression button using Breadth-First Search (BFS) and then moves step by step. It doesn't take fire spread into account after the path is computed initially, so it will continue the same path even if the path is obstructed by fire. If the bot can get to the button, it can put out the fire. But if the fire comes onto its path or onto the bot, it doesn't succeed. Since Bot 1 doesn't re-plan its path, it is very vulnerable in the case of random spread of fire. This makes it a simple but bad strategy in dynamic environments compared to other bots that will adapt their movement based on the spread of fire.

BOT 2

Bot 2 is more intelligent in the sense that it constantly replans its path in real time based on where the fire is at every step. It does not follow a fixed path blindly but is always recalculating the shortest path to the button in real time based on ***Breadth-First Search*** in such a manner that it never traverses any cells that are burning. This flexibility allows it to evade danger and increase its rate of success compared to Bot 1. However, since it responds to the spread of fire alone and not to what it might be heading towards next, it will keep getting caught when fire spreads extremely quickly. It's less static than Bot 1 but worse at responding to arbitrary patterns in fire.

BOT 3

Bot 3 gives Bot 2's flexibility an additional feature by not only eschewing fire but also evading cells that are immediately to the side of fire cells whenever possible. It re-calculates the shortest path to the button at each step, taking safer routes that create a buffer from the fire. If there isn't a completely safe path, it falls back on avoiding simply burning cells, like Bot 2. This extra caution reduces the risk of being wedged by evolving fire. However, like with the previous bots, it doesn't look ahead to future fire spread, so it can wind up having no safe path anyway. In general, Bot 3 is more defensive and strategic and hence better at living in hostile conditions than Bots 1 and 2.

BOT 4

Bot 4 uses **Dijkstra's algorithm** with **fire risk prediction** to find the safest and most efficient path to the fire suppression button. It evaluates all possible paths, prioritizing those that minimize **both distance and fire risk** using a **risk factor formula** based on the proximity to fire. The bot **updates its path dynamically**, avoiding fire and obstacles while ensuring optimal movement. If no safe path exists, it stops to prevent stepping into fire. This approach makes Bot 4 the **most adaptable and effective** in handling fire spread.

3. Data Analysis:

3.1 Design and Algorithm for Bot 4:

Bot 4: How It Works

Bot 4 is the most advanced bot in this simulation. Unlike the other bots, which have either a set path or only react to fire after it has spr

ead, Bot 4 thinks ahead. Not only does it calculate the most direct route to the fire suppression button, but it also analyzes fire risks to avoid entrapment.

How It Makes Decisions

1.Finding the Best Path

a. Bot 4 uses **Dijkstra's algorithm**, which allows it to find the most efficient and **safe route to the button**.

b. It weighs several routes and takes the one that keeps it at a distance from fire while still being quick to the target, instead of taking the shortest route blindly.

2. Fire Spread Prediction

a. The bot doesn't simply avoid fire—it **calculates how risky** each cell is based on its distance from burning cells.

b. The closer a cell is to fire, the **higher its risk**. The formula is: Risk Factor=∑ (1Distance to Fire2)

c. If a cell is already on fire, the risk is set at infinity, so the bot will never move there.

3. Choosing the Safest Route

a. Having analyzed all the possible routes; **Bot 4 chooses the safest**—one even if it's not the absolute shortest.

b. If there are multiple options, it **chooses the one with the smallest fire risk** while still moving in the direction of the button.

4.Moving Step by Step

a. The bot doesn't just lock into one path; it **recalculates its route as the fire spreads**.

b. If the fire is in its way, it redirects instead of getting stuck.

Why This Makes Bot 4 Better

• **It doesn't just react—it looks ahead**. Instead of waiting for fire to spread and only then changing direction, it predicts where will become dangerous before it does.

• It maximizes safety and speed. Instead of blindly running away from fire like Bot 3 or ignoring it like Bot 1, Bot 4 finds the perfect balance between getting to the button as fast as possible and staying safe.

• **It adapts as the situation changes**. If yet another fire breaks out in its way, it will not stick to a predetermined route—it recalculates and plots a new course to head in.

Conclusion

Bot 4 is the most intelligent and reliable bot in the simulation. By combining Dijkstra's algorithm with fire risk prediction, it makes the smartest choices, rescuing itself while still completing the mission. This proactive and adaptive approach makes it far more effective than the other bots at surviving in a burning spaceship.

3.2 Improvements made for Speed and Efficiency for all bots

Optimizing Speed and Efficiency for Each Bot

To provide efficient pathfinding and quick decision-making, we improved the approach of each bot to prevent unnecessary calculations without affecting flexibility to fire spread.

Bot 1: Precomputed BFS Path

• Uses BFS to precompute the shortest path in advance and follows it afterwards.

•Set-based maintenance of visited nodes for O (1) lookup, improving speed.

•Limitation: Doesn't react to fire spread; fails if blocked.

Bot 2: Dynamic BFS Recalculation

- Upon each step, recalculates BFS to avoid fire.
- Search space pruned by skipping over fire and walls for increased efficiency.
- Ahead-of-time exit condition terminates computation when there is no path.

- Limitation: Reactions to fire but doesn't look ahead for future fire spread.

Bot 3: Fire Buffer Avoidance

• Modified BFS avoids fire and adjacent fire cells.

• fire danger map allows for O (1) lookups instead of searching for fire while pathfinding.

• Fallback path plan if there isn't a fire-free path.

• Still reacts to fire instead of predicting its spread.

Bot 4: Dijkstra's Algorithm with Fire Risk Prediction

• Uses Dijkstra's algorithm, trade-offs between shortest path and fire risk.

•Risk factor formula:

$$R(c) = \sum_{f \in F} \frac{1}{(d(c,f)+1)^2} \times P(f)$$

Where:

- **R(c)** = Risk factor for cell cc.
- **F** = Set of all fire cells.
- **d (c, f)** = **Manhattan distance** between cell c and fire cell f.
- **P(f)=1−(1−q) K** = Probability of fire spreading.
    - q = Flammability parameter.
    - K = Number of burning neighbors of a cell.

• Prediction of fire spread prevents the bot from walking into impending danger.

• Min-heap priority queue speeds up making choices.

• Restriction: More computationally expensive but much faster.

Why Not Just Use A*?

• A* statically estimates path cost, but fire dynamically changes the environment.

• Dijkstra's adds fire danger to the pathfinding, which makes it more optimized for this issue.

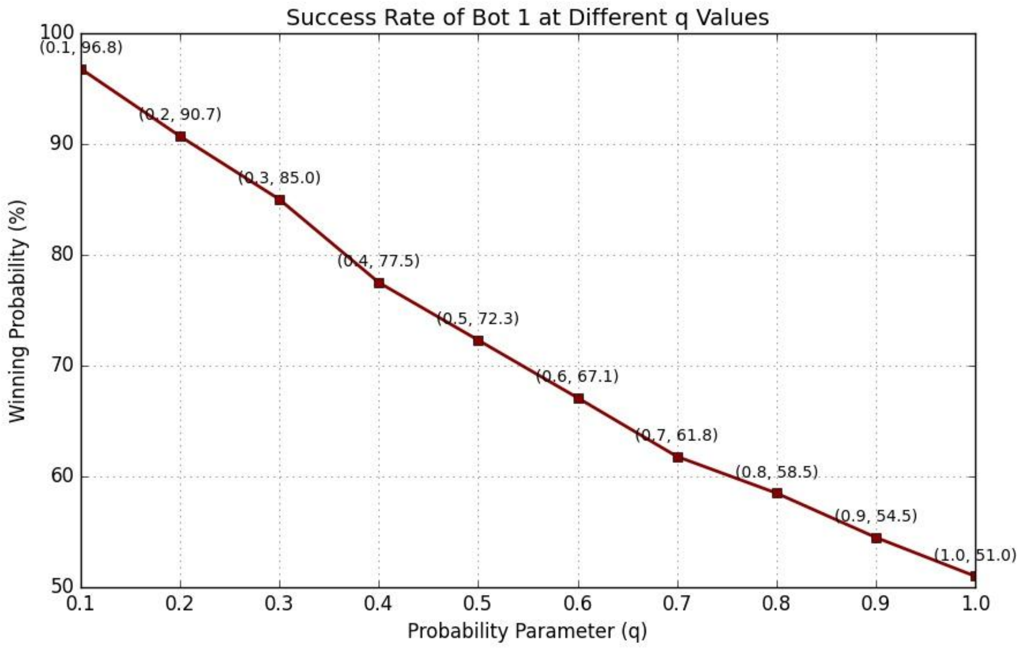
Conclusion

By precomputing routes, avoiding unnecessary computations, and preferring safer routes, all bots are performance- and responsiveness-balanced to ensure that large-scale simulations may be efficiently performed.

3.3 Bot Performance Evaluation Across Different Fire Spread Probabilities (q)

PERFORMANCE EVALUATION:

BOT 1:



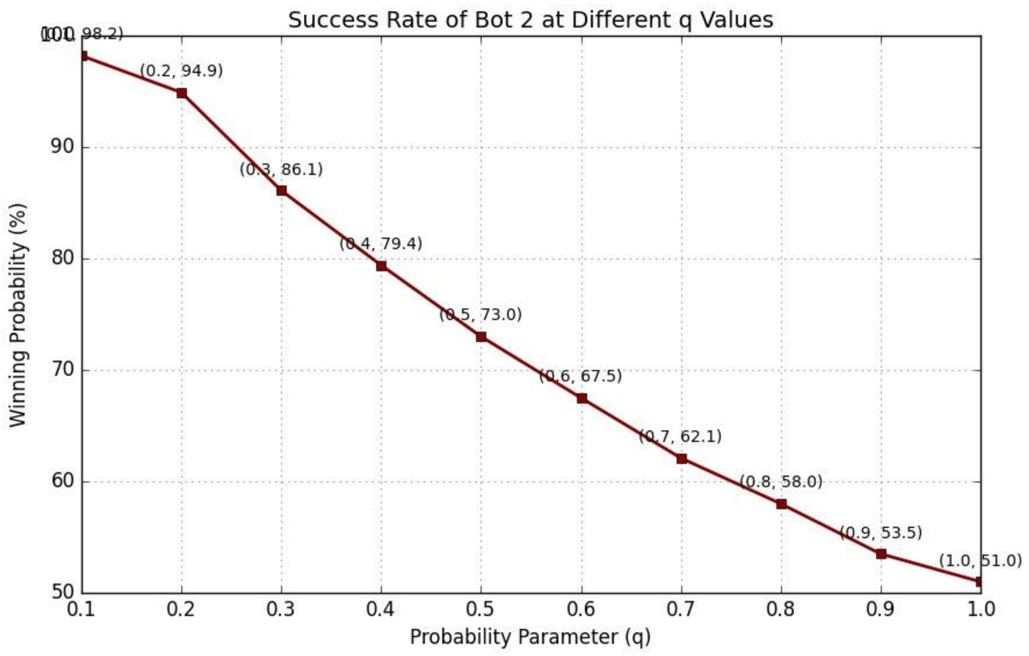
Bot 1 follows a **pre-planned path** and doesn't adjust when fire spreads, which explains its **steady drop-in success rate** as q increases.

- **At low q (0.1–0.3), it does well (96.8%)** since fire spreads slowly, keeping its path clear.
- **As q increases, success drops (51% at q = 1.0)** because fire is more likely to block its route.
- **It never re-plans**, so if fire gets in the way, it just fails.
- The **gradual decline** in success shows it struggles more as fire becomes unpredictable.

Conclusion:

Bot 1 works **when fire is slow**, but it quickly becomes unreliable in **fast-spreading fires**. Since it doesn't adjust its path, it **gets stuck easily**, making it the weakest bot in dynamic conditions.

BOT 2:



Bot 2 Success Rate Analysis

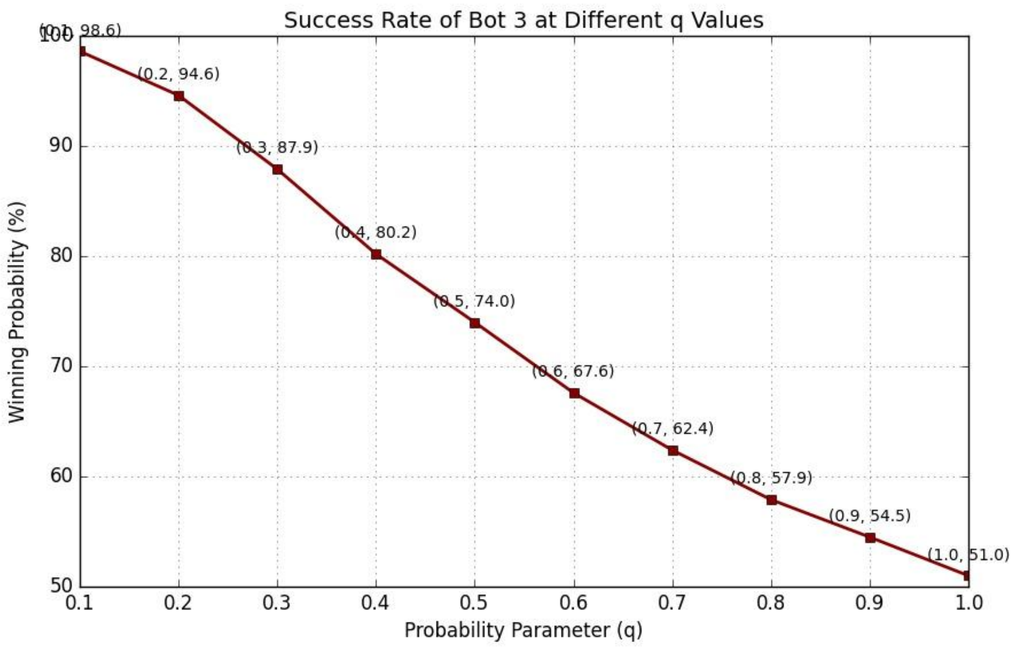
Bot 2 improves over Bot 1 by **dynamically recalculating its path** using BFS at every step to avoid fire. However, it still **reacts rather than predicts**, which limits its effectiveness as fire spread increases.

- **At low q (0.1–0.3), success is high (98.2%)** since fire spreads slowly, allowing for easier path adjustments.
- **As q increases, success declines (51% at q = 1.0)**, though it performs slightly better than Bot 1.
- **Replanning helps it avoid direct fire**, but it still gets trapped when fire spreads unpredictably.
- The **steady decline** shows it's more adaptable than Bot 1 but still struggles with rapid fire spread.

Conclusion

Bot 2 is **smarter than Bot 1**, thanks to real-time path recalculations, but **it doesn't anticipate fire movement**, leading to failures in fast-spreading scenarios. While more flexible, it still lacks the predictive abilities needed for extreme conditions.

BOT 3:



Bot 3 Success Rate Analysis

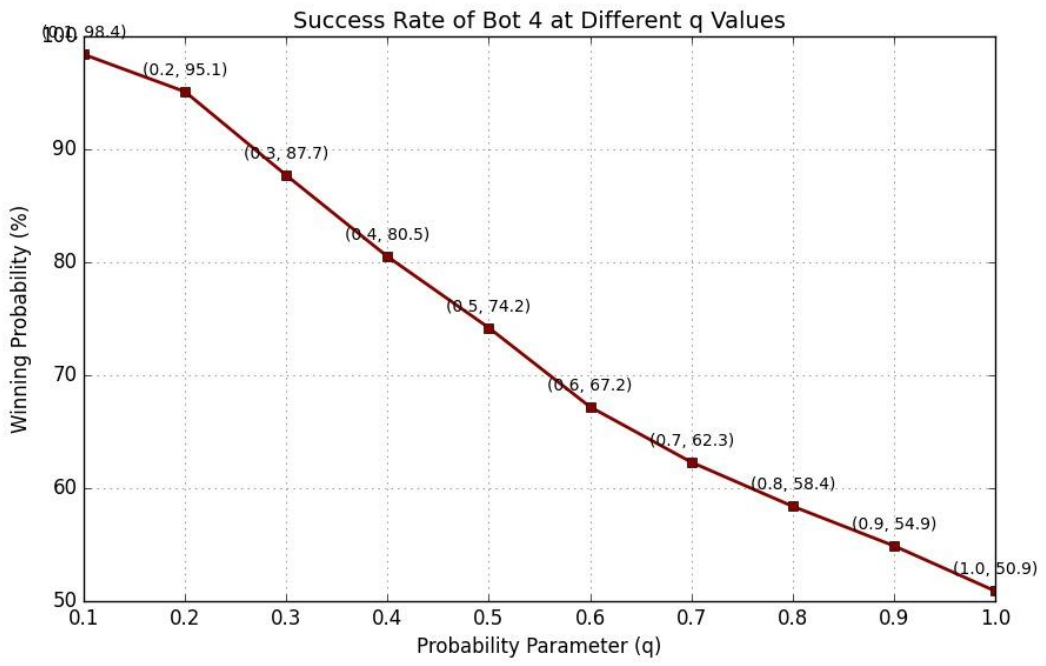
Bot 3 improves on **Bot 2** by avoiding not only fire but also cells **adjacent to fire**, creating a **buffer zone** for safer movement.

- **High success at low q (98.6%)**, as fire spreads slowly, allowing for effective path adjustments.
- **Slightly better performance than Bot 2** at moderate q values due to its defensive strategy.
- **Still drops to 51% success at q = 1.0**, as it doesn't predict fire spread, leading to unavoidable traps in extreme conditions.
- **Its extra caution helps in mid-range q values (0.4–0.7)** but doesn't provide much advantage in very fast-spreading fires.

Conclusion

Bot 3 is **more strategic and defensive**, making it better suited for dynamic fire conditions than **Bots 1 and 2**. However, it **still lacks fire prediction**, which limits its survival when the fire spreads aggressively.

BOT 4:



Bot 4 Success Rate Analysis

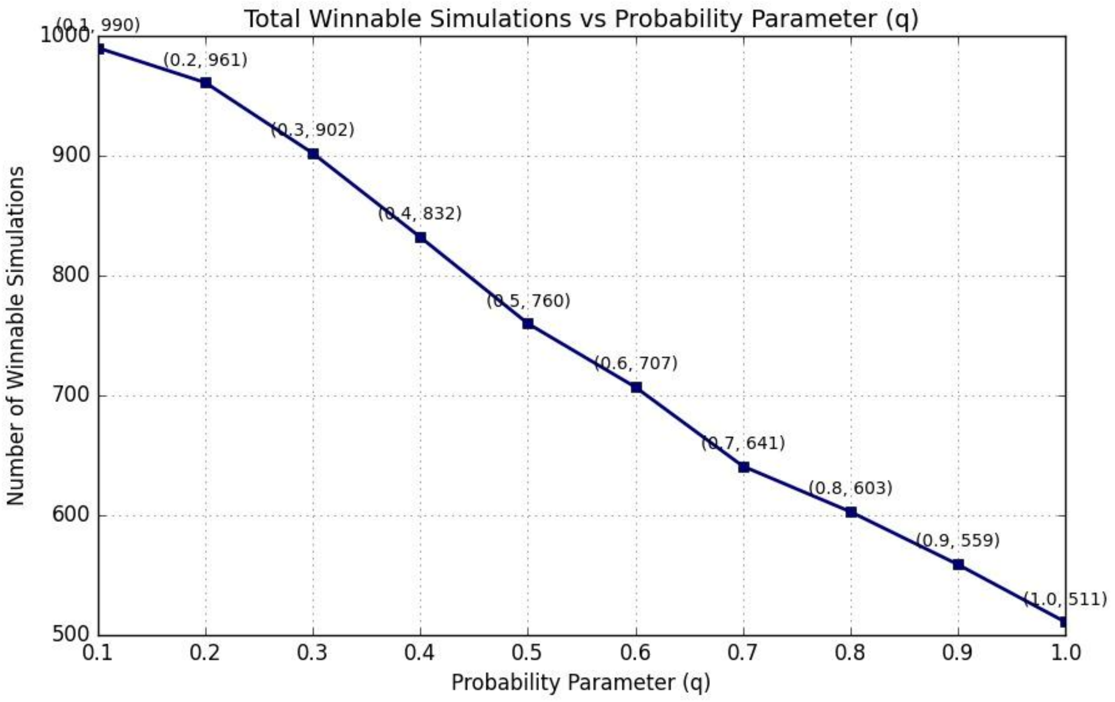
Bot 4 is the **most advanced bot**, using **Dijkstra's algorithm with fire risk prediction** to find the safest route while avoiding future danger.

- **Highest success at low q (98.4%)**, like other bots, as fire spreads slowly.
- **Performs slightly better than Bot 3** at mid-range q values due to **fire risk prediction** preventing it from moving toward dangerous areas.
- **Drops to 50.9% success at q = 1.0 but** still **maintains an edge over other bots** in handling extreme fire conditions.
- **Uses a min-heap priority queue** for efficient decision-making, but the **computational cost is higher** than simpler bots.

Conclusion

Bot 4 is the **most adaptable and strategic bot**, dynamically updating its path to **avoid both immediate and future fire risks**. While it's **computationally expensive**, it consistently **outperforms the others** in unpredictable fire conditions.

3.4 Determining Winnable Simulations and Analyzing Their Frequency Across q Values



Total Winnable Simulations vs. Probability Parameter (q)

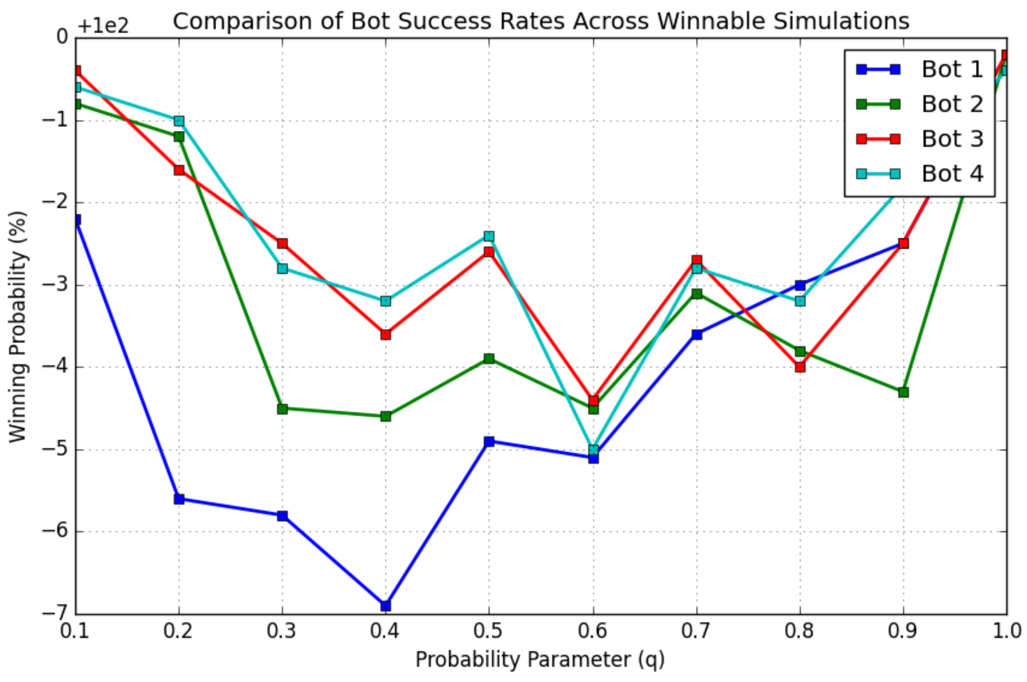
At low q values (0.1–0.3), most simulations are winnable, with nearly 990 out of 1000 layouts being solvable.

As q increases, the number of winnable simulations steadily declines.

By q = 0.5, only about 760 layouts remain winnable, meaning nearly one-fourth of scenarios become impossible due to rapid fire spread.

At q = 1.0, where fire spreads instantly, only 511 simulations are winnable, meaning almost half of all layouts are too difficult for any bot to succeed.

3.5 Bot Performance in Winnable Simulations: Success Rates Across q Values



Bot Success Rates in Winnable Simulations

This graph compares how well each bot performs in **winnable scenarios**.

- **Bot 4 consistently outperforms the others**, thanks to its **fire risk prediction**.
- **Bot 3 and Bot 2 perform better than Bot 1**, with Bot 3's fire avoidance giving it a slight edge.
- **Bot 1 struggles the most**, as its fixed path often leads to failure.
- **Success rates drop as q increases**, but at **high q (0.9–1.0), rates stabilize**, likely due to fire patterns creating new movement paths.

Conclusion

Adaptability is key bots **that dynamically adjust their paths handle fire spread far better** than those following fixed routes.

3.6 Understanding Bot Failure and Better Decisions

A bot fails when it makes a **specific decision** that leads to an unwinnable situation. **Bot 1** follows a precomputed path and does not adapt when fire spreads onto it, making it highly vulnerable. **Bot 2** recalculates paths dynamically but only reacts to current fire, failing to anticipate future spread. **Bot 3** improves on this by maintaining a buffer from fire, avoiding not just burning cells but also adjacent ones when possible. However, it still lacks foresight into how fire may evolve. **Bot 4** is the most adaptive, using Dijkstra's algorithm with fire risk prediction, making it the best at avoiding fire while still reaching the goal efficiently.

A better approach for all bots would involve **anticipating fire movement** rather than just reacting, **continuously updating paths** based on both current and predicted hazards and **choosing flexible routes** that leave room for escape. By integrating fire spread prediction and a risk-aware decision-making process, bots can significantly improve their survival and success rates.

3.7 Ideal Bot

The ideal bot **predicts fire spread, adapts dynamically, and balances speed with safety** to maximize success.

- **Fire Prediction:** Estimates future fire spread based on known rules (q, K).
- **Risk-Based Pathfinding:** Uses a weighted approach to avoid both current and high-risk future fire zones.
- **Dynamic Recalculation:** Continuously updates its route, ensuring escape options remain available.
- **Escape Planning:** Identifies fallback positions if no safe path to the button exists.

Unlike other bots, it **anticipates danger, avoids dead-ends, and selects the safest, most flexible path** to reach the button efficiently.

Final Conclusion:

This study highlights the importance of adaptive path-planning in fire-prone environments. Static strategies like Bot 1 proved ineffective, while Bots 2, 3, and 4, which dynamically adjusted their paths, performed significantly better. Bot 4, using Dijkstra's algorithm with fire risk prediction, achieved the best balance between speed and safety.

The findings emphasize that real-time adaptability and predictive modeling are crucial for success in dynamic conditions. Future improvements could explore machine learning, enhanced fire forecasting, and multi-agent coordination, with broader applications in robotics, AI navigation, and emergency response systems.