Chain Reaction - Report

2105027

June 16, 2025

Contents

1 Experimental Setup

To evaluate the performance of my AI agent, I ran a series of games using the Minimax algorithm with alpha-beta pruning. The AI played as the Red player, while the blue player was controlled by another heuristic agent.

Depth Levels: I tested the AI at different search depths: **2, 3**. Deeper levels give the AI more foresight, but increase computation time exponentially. To reduce the time, I chose only five moves inside the minimax function.

Time Environment: All tests were performed on a standard laptop with an M1 processor and 8GB RAM. The average time per move was measured using javascript date.now() from the front-end.

Game Duration: Each game continued until an AI agent won.

Evaluation Functions: The AI used five different heuristic combinations to evaluate non-terminal board states, including:

- Total orb count difference
- Number of controlled cells
- Number of positional advantages
- Explosion Probability
- Chain explosion probability

Matchups: I tested:

- Heuristic A vs Heuristic B

2 Heuristics

My AI agent evaluates each game state using five domain-specific heuristics, each reflecting different strategic priorities in Chain Reaction:

1. **Total Orb Count:** Calculates the total number of orbs owned by the player. This gives a basic idea of material advantage, but can be misleading if orbs are vulnerable to chain reactions.

2. **Controlled Cells:** Counts the number of distinct cells occupied by the player. This encourages the AI to control more of the board rather than stacking orbs in fewer locations.

3. **Positional Advantage:** Measures how many player-owned orbs are placed on corners and edges. These positions are safer because their critical mass is lower, making them harder to explode.

4. **Explosion Readiness:** Counts the number of cells that are just one orb away from reaching their critical mass. These cells are on the verge of exploding and thus indicate an offensive potential.

5. **Chain Explosion Potential:** Detects if an almost-exploding cell is adjacent to another cell that is also close to critical mass. This heuristic helps the AI identify and prepare for multi-step chain reactions.

3 Results

I conducted a series of pairwise matches between different heuristic configurations (denoted Heuristics 1 through 5). Each match reports the two heuristics involved, the time (in seconds) taken to complete the game, and the winner (the heuristic that performed better).

Heuristic A	Heuristic B	Time (s)	Winner
1	2	66.143	1
1	3	98.556	1
1	4	85.507	1
1	5	3.263	1
2	3	99.681	2
2	4	123.021	4
2	5	3.271	2
3	4	119.842	4
3	5	3.172	3
4	5	5.275	4

Table 1: Pairwise matches between heuristics showing time and winning heuristic.

4 Winning Percentages

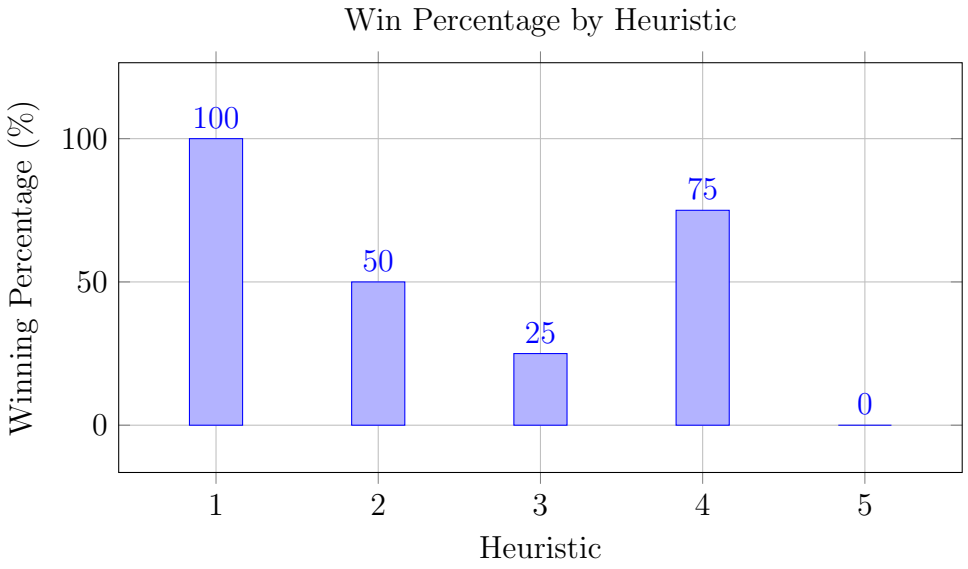
The table below summarizes the overall performance of each heuristic across all pairwise matches. It is sorted in descending order of winning percentage.

Heuristic	Wins	Winning Percentage (%)
1	4	100.00
4	3	75.00
2	2	50.00
3	1	25.00
5	0	0.00

Table 2: Heuristic performance based on win percentage.

5 Heuristic Win Rates (Visualization)

The following bar chart visually compares the winning percentages of each heuristic:



6 Analysis

From the results, it is evident that **Heuristic 1**, which primarily focuses on total orb count and board control, consistently outperformed the others in terms of winning percentage. It won all four matches against Heuristics 2, 3, 4, and 5 with moderate computation time.

Heuristic 4, which emphasizes positional safety (favoring edges and corners), also performed well in deeper-level matches, defeating Heuristics 2, 3, and 5. It took longer computation time, indicating that its strategy involved more complex state evaluations and deeper planning.

Heuristic 5, which was designed to optimize for quick chain reactions, lost all matches. While it led to very short game durations (under 5 seconds), this suggests that it prioritized premature explosions without enough board control, making it highly aggressive but strategically weak.

Trade-offs Observed:

- Simpler heuristics (like H1 and H2) resulted in faster and more consistent outcomes.
- Chain-based heuristics (like H4) led to deeper searches and longer runtimes but were effective in high-depth scenarios.
- Heuristics focusing solely on explosion readiness (like H5) may be too risky without supporting positional awareness.

These insights confirm that a balanced heuristic — combining orb advantage, cell control, and positional safety — is most effective for stable long-term performance.