Evaluation of Heuristics in Chain Reaction AI

2105166

June 16, 2025

1 Experimental Setup

The system is divided into two main components: a backend AI engine and a frontend user interface. The backend is implemented in Python and is responsible for decision-making using the minimax algorithm with alpha-beta pruning. The frontend, built using PyQt6, provides a graphical interface for visualizing the Chain Reaction game and for enabling human-vs-AI or AI-vs-AI gameplay.

Each AI agent employs one of five heuristic evaluation functions, or a random baseline, to guide its minimax search. The search depth for all agents is fixed at 3 ply to ensure fair comparisons.

Communication between the backend and frontend is handled through a shared plain text file named `gamestate.txt`. This file-based protocol encodes both the current player and the full game board state. The frontend writes the latest move to the file and waits for the AI to update the file with its move, after which the frontend reads and displays the new game state. This simple yet effective mechanism enables decoupling of the UI from the decision logic.

The board used for testing consists of 6 rows and 9 columns. For each heuristic pair, 6 games were played (3 with each agent playing Red and Blue), resulting in 30 total games per heuristic. Timing for each AI move was recorded using Python's `time.perf_counter()` to assess computational efficiency.

2 Results

Table 1 summarizes the performance statistics for each heuristic, including the number of games played, wins, losses, draws, win rate, and average time per game.

Table 1: Performance Summary of Heuristics

Heuristic	Games Played	Wins	Losses	Draws	Win Rate (%)	Avg. Time (s)
Random (H0)	30	4	26	0	13.33	0.147
Heuristic 1 (H1)	30	20	10	0	66.67	0.218
Heuristic 2 (H2)	30	14	16	0	46.67	0.247
Heuristic 3 (H3)	30	7	23	0	23.33	0.217
Heuristic 4 (H4)	30	16	14	0	53.33	0.256
Heuristic 5 (H5)	30	29	1	0	96.67	0.276

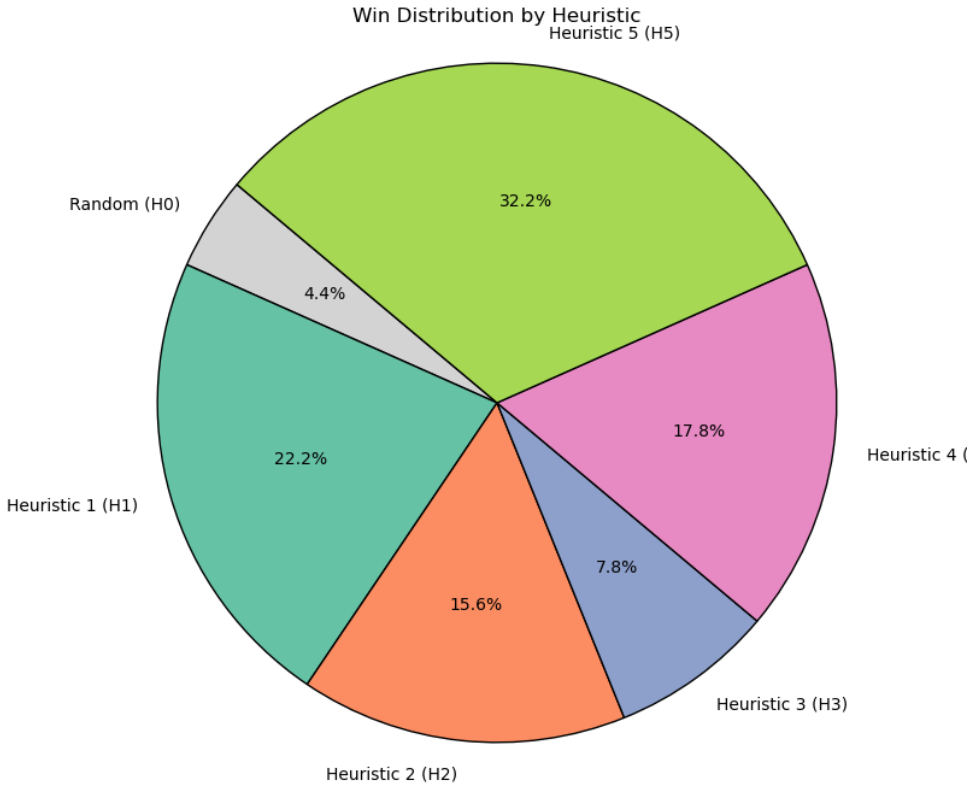

Figure 1: Win Distribution by Heuristic

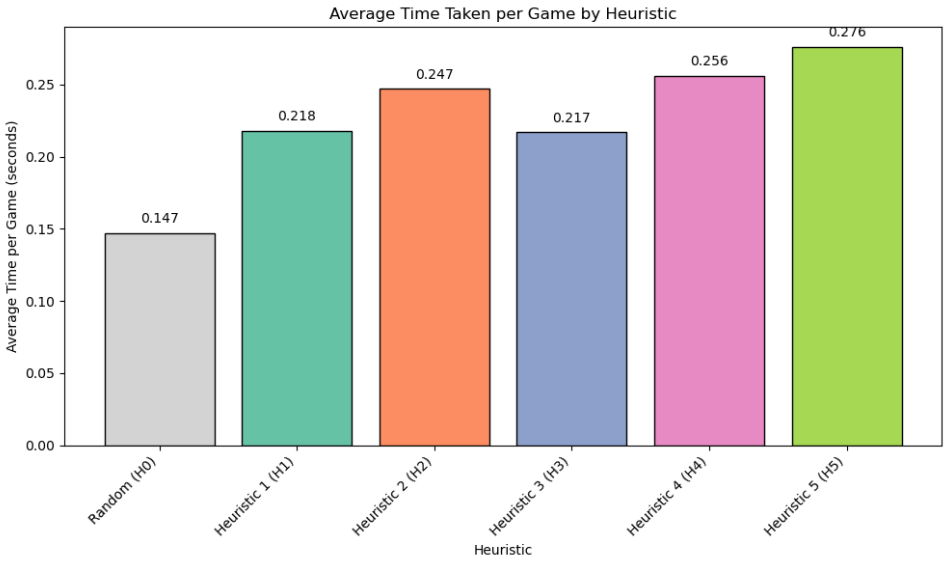

Figure 2: Average Time Taken per Game by Heuristic

3 Heuristic Details

Each heuristic function is designed to evaluate the desirability of a game state for the AI player. The five heuristics used in this study are as follows:

- **Heuristic 1 (H1) - Fitness:** This heuristic simply sums the number of orbs controlled by the AI player and adds a constant factor. It rewards states with more owned orbs, prioritizing material advantage.
- **Heuristic 2 (H2) - Stability:** This function measures how stable the AI's orbs are. A cell is considered more stable if it has fewer orbs relative to its critical mass. The heuristic penalizes positions where the AI has many unstable cells (i.e., vulnerable to opponent explosions) and rewards positions with stable ones.
- **Heuristic 3 (H3) - Threat:** This heuristic evaluates the number of immediate threats the AI poses to the opponent. It rewards configurations where opponent cells are close to exploding and are adjacent to AI-controlled cells, creating potential chain-reaction opportunities.
- **Heuristic 4 (H4) - Control:** This heuristic estimates board control based on the number of neighboring cells around each player's orbs. It rewards positions where the AI's orbs have higher spatial influence compared to the opponent, reflecting territorial dominance.
- **Heuristic 5 (H5) - Diversity:** This heuristic encourages a wider spatial distribution of the AI's orbs. It calculates the average pairwise Manhattan distance between cells owned by each player. A more dispersed setup is rewarded, as it tends to be less vulnerable to chain reactions and promotes long-term survival.

4 Discussion

The results clearly show that all heuristics outperform the random baseline, confirming the importance of strategic evaluation in the game.

Heuristic 5 (H5) demonstrated the best performance with a dominant win rate of 96.67%, indicating superior strategic decision-making. It required the most average computation time per game (0.276 seconds), suggesting a trade-off between decision quality and computational cost.

Heuristic 1 (H1) and **Heuristic 4 (H4)** also performed well, achieving win rates of 66.67% and 53.33%, respectively, with moderate computational overhead.

Heuristic 2 (H2) and **Heuristic 3 (H3)** lagged behind, with H3 showing particularly poor performance (23.33% win rate), indicating potential inefficiencies or weaknesses in their evaluation criteria.

The average time per game increases with heuristic complexity, confirming the expected trade-off: more sophisticated heuristics generally yield better play at the cost of longer computation times.

5 Conclusion

This experimental evaluation highlights that careful heuristic design significantly impacts AI performance in Chain Reaction. The best-performing heuristic (H5) provides a near-optimal balance between game outcome and computational expense. Future work may include exploring deeper search depths, hybrid heuristics, and adaptive time management to further enhance AI performance.