CSE 318: Assignment-03
Report: Adversarial Search in Chain Reaction

Mohammad Raihan Rashid
Student ID: 2105046

Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology

June 9, 2025

Contents

1 Introduction

This report details the design, implementation, and evaluation of an AI agent for the game Chain Reaction. The project explores various adversarial search techniques and heuristic evaluation functions to create an intelligent and competitive opponent. The game was built as a full desktop application, providing a graphical user interface for gameplay and configuration.

1.1 Technology Stack and Unique Approach

The project was developed using a modern, performance-oriented tech stack, chosen to explore the capabilities of emerging technologies for desktop application development.

- **Backend: Rust.** The core game logic, board representation, and all AI algorithms were implemented in Rust. This choice was motivated by Rust's promise of performance, memory safety, and its powerful type system, which are highly beneficial for building a complex and reliable game engine.
- **Frontend: SvelteKit.** The user interface was built using SvelteKit, a modern web framework known for its simplicity and high performance. It compiles components to highly efficient vanilla JavaScript, which results in a fast and responsive UI.
- **Desktop Application Framework: Tauri.** The Rust backend and Svelte frontend were bundled into a single, cross-platform desktop application using Tauri. Unlike other frameworks like Electron, Tauri leverages the operating system's native webview, resulting in a significantly smaller binary size and lower memory footprint.

A unique aspect of this project was the implementation of a flexible, modular architecture allowing for real-time configuration of the AI. The system was designed to allow users to select from multiple AI strategies and combine various heuristics on the fly, creating a powerful platform for testing and comparing different AI designs.

2 Heuristic Functions

Five distinct heuristic evaluation functions were designed and implemented to guide the AI agent's decision-making process. The goal was to explore a range of strategies, from simple material advantage to complex positional and tactical considerations.

2.1 Orb Difference

This is the most fundamental heuristic. It provides a direct measure of material advantage on the board. The score is calculated as the simple difference between the number of orbs owned by the AI and the number of orbs owned by the opponent.

$$\text{score} = \text{orbs}_{\text{AI}} - \text{orbs}_{\text{opponent}}$$

2.2 Peripheral Control

This heuristic recognizes the strategic importance of different board positions. It assigns higher values to cells that are more stable and have greater potential for initiating chain reactions. Corner cells, with a critical mass of 2, are the most valuable, followed by edge cells, and finally the less stable inner cells. The score is the sum of weighted values for all cells controlled by the AI minus the sum of weighted values for the opponent's cells.

$$w(c) = \begin{cases} 3 & \text{if } c \text{ is a corner cell} \\ 2 & \text{if } c \text{ is an edge cell (not a corner)} \\ 1 & \text{if } c \text{ is an inner cell} \end{cases}$$

$$\text{score} = \sum_{c \in \text{AI's cells}} w(c) - \sum_{c \in \text{Opponent's cells}} w(c)$$

2.3 Territory Control

This heuristic focuses on board control by simply counting the number of cells each player occupies, regardless of the number of orbs within them. The score is the difference in the number of cells controlled by the AI and the opponent. It prioritizes expansion and holding territory.

$$\text{score} = \text{count(AI's cells)} - \text{count(Opponent's cells)}$$

2.4 Chain Reaction + Conversion Potential

This is a combined tactical heuristic that assesses immediate threats.

- **Chain Reaction Potential:** It identifies "critical cells" – those that are exactly one orb away from exploding. It adds a fixed value for each of the AI's critical cells and subtracts a value for each of the opponent's critical cells.
- **Conversion Potential:** It enhances the threat analysis by looking at the neighbors of these critical cells. If an AI's critical cell is adjacent to an opponent's cell, the heuristic adds the number of orbs in that opponent's cell to the score, valuing the potential capture. Conversely, it subtracts value if an opponent's critical cell threatens to capture one of the AI's cells.

2.5 Cascade Potential

This heuristic looks beyond a single explosion and rewards setting up multi-stage chain reactions. It identifies the AI's critical cells and then evaluates the "instability" of their neighbors.

$$\text{score} = \sum_{c \in \text{AI's critical cells}} \left(\sum_{n \in \text{neighbors}(c)} (\text{orbs}(n) + \text{bonus}(n)) \right)$$

Where 'bonus(n)' is a significant extra value added if the neighboring cell 'n' is itself a critical cell, indicating a high potential for a cascading explosion. The same logic is applied to the opponent's cascade potential, which is then subtracted from the total score.

2.6 Combining Heuristics

A key feature of our system is the ability to combine multiple heuristics. The final evaluation score is a weighted sum of the scores from all selected heuristics. This allows for the creation of a powerful, nuanced game agent where, as our results show, the combined strategy consistently outperforms any single heuristic.

3 Experimental Setup

A series of experiments were conducted to evaluate the performance of the implemented heuristics.

- **Board Size:** All games were played on a board of height 9 and width 6.
- **Matches:** For each heuristic-vs-heuristic matchup, two games were played to account for any first-player advantage, with the Red and Blue teams interchanged.
- **Search Depth:** The primary search depth for all agents in the tournament was set to 3. For any matchups that resulted in a tie (1-1), a tie-breaker match was played with an increased search depth of 4.
- **Timeout:** To manage the computational complexity, especially in late-game scenarios, a maximum thinking time was implemented. If an AI agent exceeds this time limit for a single move, it returns the best move found so far at the depth it was able to complete.
- **Agents:** The tournament involved the five main heuristic agents and a 'Random' agent that selects a legal move uniformly at random.

4 Experimental Results

The following sections detail the results of the experiments.

4.1 Heuristic vs. Random Agent Performance

All developed heuristics were tested against the Random agent. As expected, every intelligent heuristic agent demonstrated a clear superiority, consistently defeating the Random agent.

Table 1: Performance of Heuristics vs. Random Agent

Heuristic (as Red)	**Winner**	**Moves**	**Time**
Peripheral Control	Peripheral Control	99	1m 14s
Orb Difference	Orb Difference	95	2m 6s
Territory Control	Territory Control	99	1m 46s
Cascade Potential	Cascade Potential	99	1m 44s
Chain Reaction+	Chain Reaction+	103	1m 15s

4.2 Heuristic vs. Heuristic Tournament

A round-robin tournament was conducted where each heuristic played against every other heuristic. The results provided surprising insights into their relative strengths.

4.2.1 Tournament Results Table

Table 2: Detailed Tournament Matchup Results

Red Player	Blue Player	Winner	Moves	Time
Peripheral Control	Orb Difference	Peripheral Control	109	2m 46s
Orb Difference	Peripheral Control	Peripheral Control	96	3m 7s
Chain Reaction+	Cascade Potential	Cascade Potential	100	6m 21s
Cascade Potential	Chain Reaction+	Cascade Potential	103	2m 56s
Territory Control	Peripheral Control	Peripheral Control	98	1m 21s
Peripheral Control	Territory Control	Peripheral Control	105	1m 47s
Orb Difference	Chain Reaction+	Orb Difference	99	2m 58s
Chain Reaction+	Orb Difference	Orb Difference	98	2m 38s
Cascade Potential	Peripheral Control	Peripheral Control	92	1m 39s
Peripheral Control	Cascade Potential	Peripheral Control	95	1m 52s
Territory Control	Orb Difference	Territory Control	95	2m 6s
Orb Difference	Territory Control	Territory Control	100	1m 57s
Chain Reaction+	Peripheral Control	Chain Reaction+	97	2m 3s
Peripheral Control	Chain Reaction+	Peripheral Control	95	1m 51s
Orb Difference	Cascade Potential	Orb Difference	97	3m 0s
Cascade Potential	Orb Difference	Cascade Potential	97	3m 36s
Chain Reaction+	Territory Control	Chain Reaction+	97	2m 11s
Territory Control	Chain Reaction+	Chain Reaction+	98	2m 40s
Territory Control	Cascade Potential	Territory Control	101	1m 36s
Cascade Potential	Territory Control	Cascade Potential	105	3m 17s

4.2.2 Tie-Breaker Matches

A three-way tie in the initial standings between Territory Control, Orb Difference, and Chain Reaction+ necessitated a round-robin tie-breaker tournament at a higher search depth of 4.

4.2.3 Final Tournament Points Table

After incorporating the tie-breaker results, the final standings were determined.

Table 3: Tie-Breaker Match Results (Depth 4)

Red Player	**Blue Player**	**Winner**	**Moves**	**Time**
Territory Control	Orb Difference	Orb Difference	96	7m 44s
Orb Difference	Territory Control	Orb Difference	101	4m 15s
Chain Reaction+	Territory Control	Chain Reaction+	101	3m 11s
Territory Control	Chain Reaction+	Chain Reaction+	93	4m 30s
Orb Difference	Chain Reaction+	Chain Reaction+	101	6m 6s
Chain Reaction+	Orb Difference	Orb Difference	-	-

Table 4: Final Tournament Standings

Heuristic	**Wins**	**Losses**
Peripheral Control	7	1
Cascade Potential	6	2
Orb Difference	4	4
Chain Reaction + Conversion Potential	3	5
Territory Control	2	6

5 Analysis of Results

5.1 Heuristic vs. Heuristic Tournament Analysis

The round-robin tournament provided a clear hierarchy of heuristic effectiveness. The heatmap in Figure 1 visualizes the head-to-head outcomes from the Red player's perspective.

Peripheral Control's Strong Performance: This heuristic emerged as the clear champion of the tournament. Its high affinity for capturing and holding corner and edge cells proved to be a dominant strategy. Besides the inherent stability of these positions, this strategy technically encircles the opponent's orbs. In many games, this led to a state where any chain reaction initiated by the agent would sweep across the board from one end to the other, a massive advantage in the endgame. It consistently outperformed smarter opponents that could see further ahead but did not prioritize this crucial positional play.

Territory Control's Weakness: This heuristic's failure stems from its simple thinking: it always tries to grab the most territory. While this can be useful in the late game, it often fails to build a strong, chain-reaction-capable base in the early and mid-game. It would lose its footing at the start and struggle to recover against heuristics that focused more on tactical capturing. Interestingly, when the AI's thinking time was limited, it performed on par with Orb Difference and Chain Reaction Potential, but failed against them when given more depth and time to think.

5.2 Cyclical Win/Loss Outcomes

The tournament revealed some surprising cyclical relationships, a "rock-paper-scissors" effect among the heuristics, particularly visible in the initial non-tie-breaker games:

- **Territory Control** defeated **Orb Difference**.

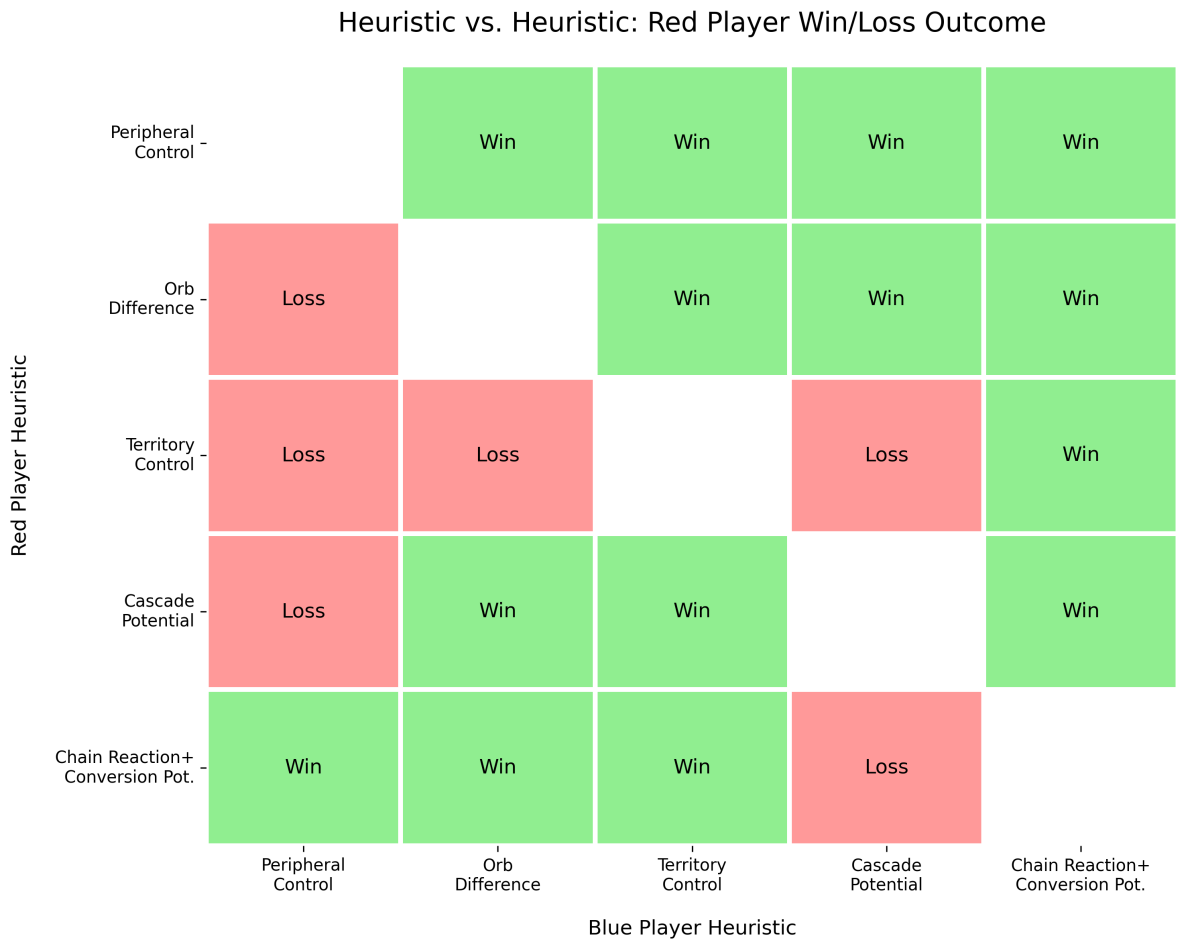

Figure 1: Win/Loss outcomes for heuristic vs. heuristic matchups. Green indicates a win for the Red Player Heuristic, while Red indicates a loss.

- **Orb Difference** defeated **Chain Reaction + Conversion Potential**.
- **Chain Reaction + Conversion Potential** defeated **Territory Control**.

This cycle demonstrates that no single simple strategy is universally dominant. Territory Control's focus on expansion beats Orb Difference's focus on raw numbers, but it leaves it vulnerable to the tactical explosions prioritized by Chain Reaction Potential.

5.3 Overall Win Rate

The overall win rate across all tournament games provides a clear hierarchy of strategic effectiveness, as shown in Figure 2.

5.4 Runtime Analysis

The runtime was analyzed in games against the Random agent, as its computation time is negligible. The time taken is therefore dominated by the heuristic agent's search time. Figure 3 shows the average game time for each heuristic when playing as Red and Blue. It is important to note that this does not reflect the true time complexity of the heuristics

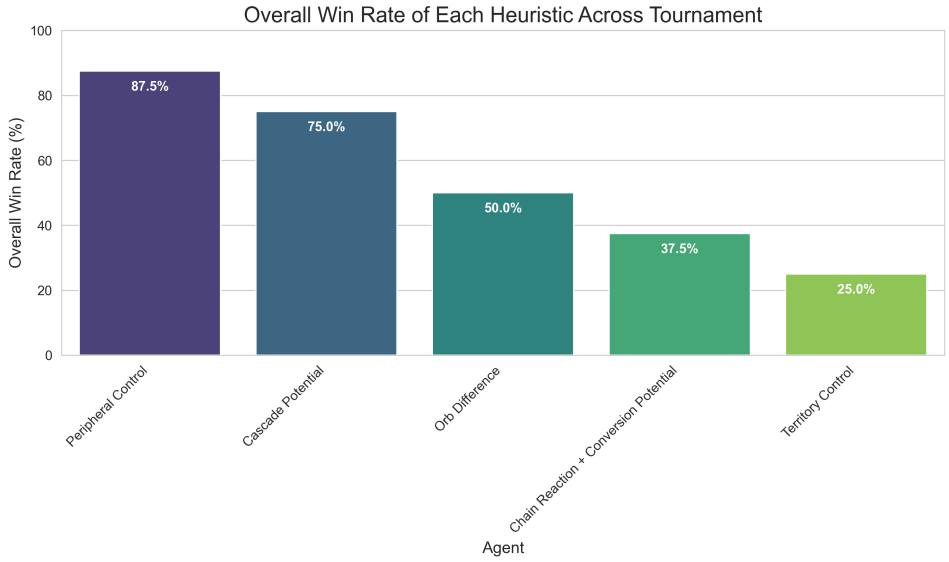

Figure 2: Overall win rate of each heuristic across all tournament games.

themselves. A shorter game time could indicate a more dominant agent that wins in fewer moves, rather than a less computationally intensive heuristic.

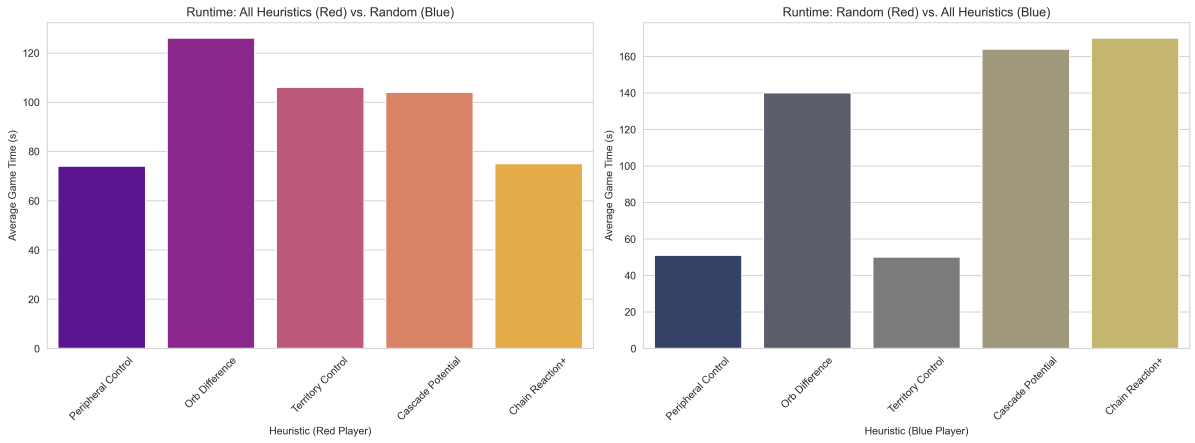

Figure 3: Runtime comparison for all heuristics vs. the Random Agent.

6 Conclusion

This project successfully developed a feature-rich Chain Reaction game with a configurable and highly capable AI opponent. The exploration of different heuristics provided deep insights into the game's strategic nature, demonstrating that a well-balanced approach that values positional control, like the 'PeripheralControl' heuristic, can outperform more narrowly focused tactical strategies.

However, the choice of the technology stack—Rust, Tauri, and Svelte—presented significant challenges for game development. While powerful, the ecosystem proved to be unreliable for this use case. We encountered numerous development hurdles:

- **Instability:** The application would randomly crash for certain game states, and at one point, was even able to crash the VS Code development environment, suggesting deep-seated issues in the framework's stability for this type of application.
- **Race Conditions:** The interaction between the Rust backend and the Svelte frontend, especially when dealing with rapid state updates, led to complex race conditions that required careful management with Mutexes and event listeners.
- **Performance Bottlenecks:** In the endgame, the sheer number of possible moves and the complexity of the resulting chain reactions caused the AI's thinking time to increase exponentially. This led to UI freezes and a lack of smooth performance. To mitigate this, we had to implement a timeout for AI moves, forcing it to return the best move found within a certain time limit, even if it hadn't completed its full search. This was a necessary compromise for playability.
- **Development Environment Issues:** The file watcher in Tauri's development server would often restart the entire backend when a log file was updated, forcing workarounds like placing log files outside the project's source directory.

While this project was an excellent learning experience for understanding Rust, Svelte, and the complexities of game AI, the chosen tech stack, particularly Tauri, proved to be unreliable for a production-quality game at this time. The manual testing required due to the framework's instability highlights these limitations.