CSE 318: Assignment-03

Report: Adversarial Search in Chain Reaction

Md. Abrar Jahin
Student ID: 2105055
Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology

June 16, 2025

Contents

1 Introduction

This report presents the design, implementation, and empirical evaluation of an AI agent developed to play the strategic board game *Chain Reaction*. Leveraging adversarial search algorithms, particularly the Minimax strategy with Alpha-Beta pruning, the agent is guided by a diverse set of domain-inspired heuristic evaluation functions. The objective of this work is to explore how different heuristics influence gameplay strength and decision-making behavior in a deterministic, turn-based environment.

2 Heuristic Functions

Nine different heuristic evaluation functions were designed and tested. Each of them attempts to model different strategic approaches to playing Chain Reaction. Each heuristic below describes its rationale, the method of calculation, and a summary of its observed behavior and performance in gameplay.

2.1 EqualEval

Rationale: Provides a baseline behavior for benchmarking. No actual evaluation logic.
It simply chooses the first cell it can find. Evaluating each cell the same.
Calculation:

$$\text{score} = 1$$

Summary: Extremely simple, returns constant value.
Behavior: Very deterministic, always chooses top-left. When depth increases, surprisingly better than some other heuristics.
Performance: Not challenging for human vs AI; predictable but fast.

2.2 MaxNearExplosionsEval

Rationale: Prioritizes cells that are on the verge of exploding, to maximize potential chain reactions.

$$\text{score} = \sum_{c \in \text{player}} \left[\delta(c) \cdot \begin{cases} 200, & \text{if crit}(c) = 2 \\ 150, & \text{if crit}(c) = 3 \\ 100, & \text{if crit}(c) = 4 \\ 0, & \text{otherwise} \end{cases} + \frac{1}{\text{orbs}(c)} \right]$$

Where $\delta(c) = 1$ if $\text{orbs}(c) = \text{crit}(c) - 1$, otherwise 0.

Summary: Rewards cells one orb away from explosion.

Behavior: Encourages explosive setups, especially with additional weightings. Although it does not explicitly optimize for deep chain explosions, adding extra weight to near-exploding cells improves tactical sharpness. Its effectiveness increases significantly at greater search depths.

Performance: At lower depths, it performs short-sightedly and misses longer-term setups. However, at depths like 5, it consistently makes strong moves and produces entertaining AI vs AI matches. It runs slightly slower than EqualEval due to additional computation but remains efficient.

2.3 LeastAttackingEval

Rationale: Reduces exposure to enemy-adjacent cells, favoring defensively safe placements. This heuristic evaluates how many of the player's cells are directly threatened by nearby opponents, and avoids overly aggressive expansion. It is especially suitable for defensive gameplay.

Calculation:

$$\text{score} = -1 \times \text{number of enemy-adjacent cells owned by player}$$

Summary: Penalizes cells surrounded by enemies.

Behavior: Extremely defensive and slightly unpredictable; safe play. Tends to create a more reserved and cautious AI that avoids risky plays. This also places orbs in a diagonal manner as expected, although it doesn't correlate to winning performances all the time.

Performance: Works best with deeper searches (depth 5 or higher). At lower depths, it underperforms due to shortsightedness. Slightly slower than `MaxNearExplosionsEval`, but provides enjoyable challenge when playing against. Not very engaging to watch in AI vs AI matches due to passive behavior.

2.4 MaxOrbsEval

Rationale: Seeks to maximize the total number of orbs controlled by the player, representing raw material advantage.
Calculation:

$$\text{score} = \sum_{c \in \text{player}} \text{orbs}(c)$$

Summary: Maximizes total orbs on the board.
Behavior: Greedy and expansionist; tends to spread across the board without prioritizing explosions or attacks. This heuristic ignores positional threats or board volatility and purely focuses on accumulating orbs.
Performance: Games take significantly longer, often requiring the agent to fill the board before triggering explosions. Interesting to analyze but not very exciting to watch due to its passive and prolonged strategy.

2.5 MaximizeOrbDifference

Rationale: Prioritizes overall material advantage by computing the net orb count difference between the player and the opponent. This simple yet effective heuristic captures immediate strength but does not consider positional or tactical factors.

Calculation:

$$\text{score} = \sum_{c \in \text{player}} \text{orbs}(c) - \sum_{c \in \text{opponent}} \text{orbs}(c)$$

Summary: Rewards higher net orb count compared to opponent.

Behavior: Realistic but slightly shortsighted; tends to be more passive early on and takes time before building up any chain reactions.
On smaller boards, the behavior becomes more observable as the agent delays stacking orbs until it gains a clear advantage. Provides a legitimate challenge for human players and improves significantly at higher depths. With depth 5, it becomes strategically engaging.

Performance: Longest runtime among the heuristics prior to this due to the depth and broad evaluations it performs each turn.

2.6 BigExplosionPotential

Rationale: Encourages a more aggressive and explosive style of play by rewarding cells that are close to critical mass, especially those clustered together or located near the center of the board.
The aim is to build up controlled potential for massive cascading reactions. This approach complements defensive heuristics like `LeastAttackingEval` by taking the opposite stance.
Calculation:

$$\text{score} = \text{critical bonus} + \text{cluster bonus} + \text{center weight bonus}$$

Where:

- **Critical bonus:** +50 for each cell with orb count one less than its critical mass.
- **Cluster bonus:** +25 for each adjacent near-exploding cell.
- **Center weight bonus:** Inversely proportional to Manhattan distance from board center. Since cells near the middle of the board tend to have higher impact.

Summary: Favors setups near explosion and clusters with central control.

Behavior: Starts with safe placements and builds up explosion potential. More aggressive than `LeastAttackingEval`, but still calculated in the early game.

Performance: Effective at building large combos, though somewhat delayed. Not very exciting to play against or watch, but strategically solid.

2.7 VulnerabilityPenaltyEval

Rationale: This heuristic penalizes player's cells that are in immediate danger of being captured by neighboring enemy cells close to explosion. It aims to reduce the risk of sudden chain reactions against the agent by discouraging vulnerable positioning. The penalty is weighted by the explosiveness of the enemy cell — the lower its critical mass, the higher the risk and penalty.
Calculation:

$$\text{score} = -1 \times \sum_{c \in \text{Player}} \sum_{n \in \text{Adj}(c)} \begin{cases} 7 - \text{crit}(n), & \text{if owner}(n) \neq \text{Player and orbs}(n) = \text{crit}(n) - 1 \\ 0, & \text{otherwise} \end{cases}$$

Summary: Penalizes board states where the player's orbs are adjacent to enemy cells about to explode.
Behavior: Highly cautious; avoids positions near volatile enemies, especially near corners and edges where chain reactions are easier.
Performance: Defensive by design, it helps preserve long-term stability and is especially useful when trying to protect a lead. However, it may underperform without tweaks to the implementation.

Comparison with LeastAttackingEval: While both heuristics emphasize defensive positioning, `VulnerabilityPenaltyEval` is more tactical and short-term focused. It reacts specifically to cells that are on the verge of exploding, whereas `LeastAttackingEval` more generally avoids any adjacency to opponents, regardless of threat level. The former prevents immediate threats; the latter avoids conflict zones altogether.
`VulnerabilityPenaltyEval`, acts more strategically than `LeastAttackingEval`, which is more naive than strategic.

2.8 OrbCapturePotentialEval

Rationale: This heuristic promotes aggressive plays by identifying cells that are one orb away from explosion and surrounded by enemy cells. It encourages placing orbs in positions that could immediately trigger a capture cascade. The more enemy-adjacent cells a nearly-exploding cell has, the higher the score.
Calculation:

$$\text{score} = \sum_{\substack{c \in \text{player} \\ \text{orbs}(c) = \text{crit}(c) - 1}} 10 \times \{n \in \text{Adj}(c) \mid \text{owner}(n) \neq \text{player}\}$$

Summary: Prioritizes near-exploding player cells that are adjacent to enemy cells.
Behavior: Tends to seek out explosive and aggressive opportunities to flip control. More tactical and immediate than positional heuristics.
Performance: Effective at pressuring the opponent and creating localized disruptions. Its behavior becomes sharper at higher depths where chain potentials are evaluated more precisely.

2.9 PositionalControlEval

Rationale: This heuristic rewards control over structurally advantageous areas of the board. Corners and edges are prioritized due to their lower critical mass and defensive strength. It assigns different weights to cells based on their position (corner, edge, or center) and multiplies these by the orb count to compute the overall score.
Calculation:

$$\text{score} = \sum_{c \in \text{player}} w(c) \times \text{orbs}(c)$$

where

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

Summary: Rewards controlling corners, edges, and maintaining high orb density in strong positions.
Behavior: Encourages stable expansion into low-risk, defensible regions. Tends to build secure territories before engaging aggressively.
Performance: Performs well in matches requiring steady control and late-game strength. Particularly effective at holding the board and avoiding chaotic blowouts. However, may not capitalize quickly on explosive opportunities.

3 Experimental Setup

To evaluate the performance of the various heuristics, a large-scale series of head-to-head matchups were conducted under controlled conditions. Unlike full-sized boards, a reduced board of size 7×4 was used to accelerate match duration and observation cycles, as games generally do not become strategically rich until the halfway point.

- **Agents:** All core heuristics described in Section 2 were used, along with a random agent baseline.
- **Board Size:** All experiments were conducted on a 7×4 board.
- **Tournament Setup:** Each AI plays Random 3 times. The winner gets to play knockout. First player bias will be eliminated through switching.
  Winners get to play in the knockout stage.
  Best performer gets to skip quarter-finals.
- **Search Depth:** Depth-based performance comparisons were performed using depths 3 and 5.
- **Timeout:** A time limit of 5 seconds per move was enforced. If an AI agent exceeded this time, a random legal move was chosen and executed in its place.

4 Heuristic vs. Random Agent Performance

Each heuristic agent was evaluated against the Random agent with a search depth of 5. As expected, most AI agents consistently outperformed the random baseline, establishing a strong strategic advantage even on the smaller 7×4 board, if their strategy was good enough.
The lack of endgame awareness also played a role in the losses of some heuristics like Orb Difference.
Each AI plays Random 3 times. The winner gets to play knockout.

Heuristic	**Winner(Best of 3)**	**Avg Moves**
EqualEval	EqualEval	25
MaxNearExplosionsEval	MaxNearExplosionsEval	26
LeastAttackingEval	Random	26
MaxOrbsEval	MaxOrbsEval	26
MaximizeOrbDifference	MaximizeOrbDifference	27
BigExplosionPotential	BigExplosionPotential	25
VulnerabilityPenaltyEval	VulnerabilityPenaltyEval	24
OrbCapturePotentialEval	Random	25
PositionalControlEval	PositionalControlEval	26

5 Knockout Tournament Based on Random Match Performance

The seven heuristic agents that defeated the random baseline participated in a single-elimination knockout tournament. The top performer (based on shortest win duration) received a first-round *bye*.

Depths will be 3 and 5. Each heuristic plays their opponent twice.

6 Knockout Tournament: Best of 3 Format

The seven best-performing heuristics from the AI vs Random matches competed in a knockout tournament. Each matchup was played as a best-of-3 series. The fastest winner by average moves (VulnerabilityPenaltyEval) received a first-round bye.
We will allow tiebreaker. Tie-breakers will happen on a 7×4 board as well as a 9×6 board.

Round [AI (depth)]	**Matchup (Best of 2)**	**Winner - Moves (W - L)**
Quarterfinal 1	EqualEval(5) vs MaxOrbsEval(3) EqualEval(3) vs MaxOrbsEval(5)	EqualEval(5) - 28-28 EqualEval(3) - 33-32
Quarterfinal 2	NearExplosionsEval(3) vs ExplosionPotential(5) NearExplosionsEval(5) vs ExplosionPotential(3)	ExplosionPotential(5) - 33-33 ExplosionPotential(3) - 31-30
Quarterfinal 3	PositionalControl(3) vs OrbDifference(5) PositionalControl(5) vs OrbDifference(3)	OrbDifference(5) - 27-27 OrbDifference(3) - 30-29
Quarterfinal 4	VulnerabilityPenalty	*(Bye)*
Semifinal 1	EqualEval(5) vs ExplosionPotential(3) EqualEval(3) vs ExplosionPotential(5)	EqualEval(5) - 31-31 EqualEval(3) - 28-26
Semifinal 2	OrbDifference(5) vs VulnerabilityPenalty(3) OrbDifference(3) vs VulnerabilityPenalty(5)	OrbDifference(5) - 25-25 OrbDifference(3) - 27-26
Final	EqualEval(3) vs OrbDifference(5) EqualEval(5) vs OrbDifference(3)	**OrbDifference(5) - 31-31** **EqualEval(5) - 27-27**
TieBreaker 1 (Red EqualEval)	EqualEval(5) vs OrbDifference(5)	**OrbDifference(5) - 31-30**
TieBreaker 2 (Red OrbDifference)	EqualEval(5) vs OrbDifference(5)	**EqualEval(5) - 30-30**
TieBreaker 3 (9x6 Board, Red EqualEval)	EqualEval(5) vs OrbDifference(5)	**EqualEval(5) - 71-70**
TieBreaker 4 (9x6 Board, Red OrbDifference)	EqualEval(5) vs OrbDifference(5)	**OrbDifference(5) - 75-74**
TieBreaker 5 (9x6 Board, Depth 6, No more heed given to player order0	EqualEval(6) vs OrbDifference(6)	**OrbDifference(6) -91-90**
Winner		OrbDifference

7 Analysis of Knockout Tournament Performance

The knockout phase provided a structured environment to assess how heuristics perform in direct competition. The results reflect not only strategic merit but also the impact of search depth, evaluation stability, and implementation details.

7.1 EqualEval's Consistent Performance

- `EqualEval`, a heuristic with no evaluation logic, reached the final and forced multiple tiebreakers.
- Its consistent, uninfluenced decision-making helped avoid blunders caused by skewed evaluations.
- The absence of weighting errors or prioritization conflicts may have contributed to its reliability under pressure.

7.2 OrbDifference's Adaptability

- `OrbDifference` improved across rounds, performing better as depth and board size increased.
- It scaled effectively in deeper searches by maintaining orb advantage over time.
- Its stable performance in extended matches led to its eventual tournament victory.

7.3 Situational Heuristics

- Heuristics like `ExplosionPotential` and `PositionalControl` relied on specific board conditions.
- They performed well when opponents created clustered or passive positions.
- Against adaptive or neutral strategies, these heuristics struggled to apply their intended advantages.

7.4 Depth Impact

- Agents configured with higher depth often outperformed their lower-depth counterparts.
- Depth 5 and 6 configurations resulted in fewer tactical errors and improved stability.
- The final tiebreaker at depth 6 on a 9×6 board ended with a one-move difference, emphasizing that depth increased foresight but did not eliminate close results.
- Overall, deeper search enhanced consistency, but effectiveness still depended on the underlying heuristic.

7.5 Implementation Considerations

- Several heuristics used hardcoded multipliers (e.g., 10, 50, 200), which could dominate the decision function.
- Lack of normalization between terms caused some heuristics to heavily bias certain move types.
- Simpler agents, like `EqualEval`, avoided these issues, benefiting from uniform behavior.

- Potential indexing or logic inconsistencies may have skewed evaluations for geometry-based heuristics.
- Minor code-level inaccuracies, though subtle, may have influenced tight matchups.

8 Conclusion

This evaluation compared several heuristic approaches to Chain Reaction using both random baselines and structured knockout tournaments. Key observations include:

- Complex heuristics are not always superior—simple and consistent agents like `EqualEval` performed reliably under competitive conditions.
- Heuristics that rely on specific opponent behaviors or positional patterns show inconsistent performance across varied matchups.
- Search depth improves decision quality, but its impact depends on the evaluation function's stability.
- Implementation details, such as weighting imbalance or indexing precision, can significantly affect strategic behavior.

Overall, success depended less on raw complexity and more on balanced scoring, adaptability, and robustness across scenarios.