CentraleSupélec | université PARIS-SACLAY

MScAI
Master of Science in Artificial Intelligence
Applied to society

AI Challenge: Vampires vs. Werewolves

Academic Year: 2024–2025

Realised by:

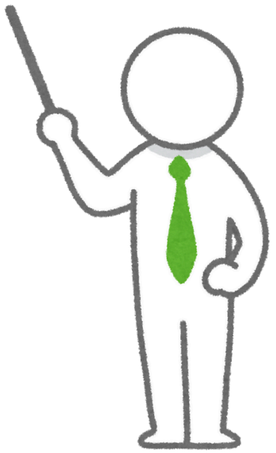
Manon LAGARDE
Maria KONTARATOU
Chaimae SADOUNE

Supervised by:

Jean-Philippe POLI

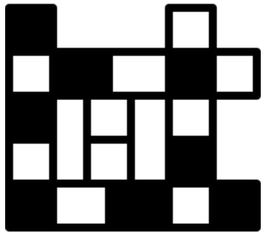
Presentation outline

- *Game Overview*
- *Game Rules*
- *Search Algorithm Used*
- *Evaluation Function*
- *Implementation & Test Results*
- *Limitations and Future Prospects*



Game Overview:

This is a turn-based strategy game where vampires and werewolves compete for dominance. The goal is to convert humans, eliminate opponents, and control as much of the grid as possible.



Game Grid

- **Humans** : Neutral characters that can be converted into vampires or werewolves

- **Vampires and Werewolves** : The two competing species, each controlled by a player

- **Empty cells :** Neutral spaces that can be traversed

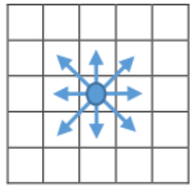
Objective

The primary goal is to maximize control of the grid while reducing the opponent's influence. Success relies on strategic planning, efficient resource management, and tactical decision-making to outnumber and overpower the adversary.

Game Rules

Movements:

- Creatures can move to adjacent cells in any of the 8 directions.
- Groups can be split into smaller units to optimize strategies and cover more ground.



Conversions and Battles:

- Humans in a cell are converted if the number of creatures moving into the cell meets or exceeds the number of humans present.
- Battles between opposing creatures are resolved probabilistically, depending on the ratio of attackers to defenders.

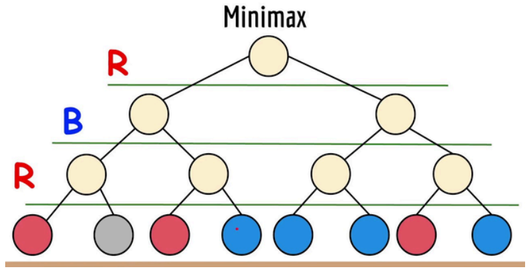
End of the Game:

- The game ends under one of the following conditions:
- The time limit is reached, and the species with the larger population wins.
- One species is completely eliminated.
- One species controls the majority of the grid.

Search Algorithm

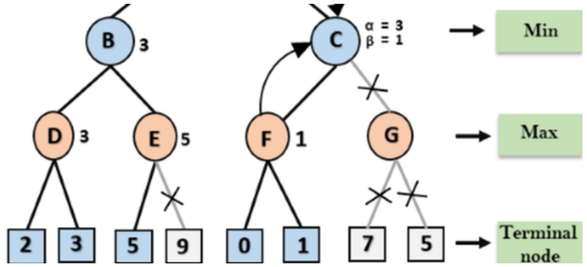
Mini-Max Algorithm

It is a recursive or backtracking algorithm which is used in decision-making and game theory. It provides an optimal move for the player assuming that opponent is also playing optimally.



Alpha-Beta Pruning

Alpha-Beta Pruning optimizes the Mini-Max Algorithm by reducing the number of nodes explored by skipping branches that cannot influence the final decision.



Depth Limitation

To manage computational complexity, the decision tree is explored up to a predefined depth, ensuring that decisions are computed within the 2-second time constraint.

Evaluation Function

The evaluation function in this project assesses the desirability of game states by calculating a weighted score based on several dynamic metrics. These metrics reflect the AI's ability to dominate the game by converting humans, defeating opponents, and positioning itself strategically. The formula for the heuristic score is :

$$\begin{aligned}\text{Evaluation Score} = {} & 10 \cdot (U_{\text{ai}} - U_{\text{op}}) + w_{\text{resource}} \cdot (\text{Resource Gain}_{\text{ai}} - \text{Resource Gain}_{\text{op}}) + \\ & w_{\text{attack}} \cdot (\text{Attack Power}_{\text{ai}}) + \text{Strategic Positioning} + \text{Distance Penalty} + \text{Random Bias}\end{aligned}$$

Advantage in Unit

Rewards game states where the AI outnumbers the opponent

$$\text{Advantage Units} = 10 \cdot (\text{Our Units} - \text{Opponent Units})$$

Resource Gain

The **Resource Gain** for the Position evaluates the potential benefit of converting humans within a specific range R from the AI's current position. It selects the position with the highest efficiency E(i, j), ensuring that the AI prioritizes the most beneficial move.

$$G = \max_{(i,j) \in \text{Range}(R)} E(i,j)$$

$$E(i,j) = \frac{H(i,j)}{d((x,y),(i,j))} \longrightarrow d((x,y),(i,j)) = \max(|x-i|, |y-j|)$$

$$\text{Resource Gain Difference} = 20 \cdot (\text{Potential Resource Gain}_{\text{us}} - \text{Potential Resource Gain}_{\text{opponent}})$$

Evaluation Function

The evaluation function in this project assesses the desirability of game states by calculating a weighted score based on several dynamic metrics. These metrics reflect the AI's ability to dominate the game by converting humans, defeating opponents, and positioning itself strategically. The formula for the heuristic score is :

$$\text{Evaluation Score} = 10 \cdot (U_{\text{ai}} - U_{\text{op}}) + w_{\text{resource}} \cdot (\text{Resource Gain}_{\text{ai}} - \text{Resource Gain}_{\text{op}}) + w_{\text{attack}} \cdot (\text{Attack Power}_{\text{ai}}) + \text{Strategic Positioning} + \text{Distance Penalty} + \text{Random Bias}$$

Strategic Positioning

Encourages advantageous placement relative to opponents, using Chebyshev distance

$$\text{Strategic Positioning} = \frac{50}{\text{Distance to Opponent} + 1}$$

Attack Power

The **Attack Potential** metric evaluates the AI's ability to eliminate opponents (vampires or werewolves) within a specific range. This evaluation helps the AI prioritize targets based on feasibility and strategic value.

$$\text{Attack Power} = \sum_i \frac{\text{Enemy Units}_i}{\text{Distance}_i}$$

Evaluation Function

The evaluation function in this project assesses the desirability of game states by calculating a weighted score based on several dynamic metrics. These metrics reflect the AI's ability to dominate the game by converting humans, defeating opponents, and positioning itself strategically. The formula for the heuristic score is :

$$\text{Evaluation Score} = 10 \cdot (U_{\text{ai}} - U_{\text{op}}) + w_{\text{resource}} \cdot (\text{Resource Gain}_{\text{ai}} - \text{Resource Gain}_{\text{op}}) + w_{\text{attack}} \cdot (\text{Attack Power}_{\text{ai}}) + \text{Strategic Positioning} + \text{Distance Penalty} + \text{Random Bias}$$

Distance Penalty

Penalizes states where allied groups are dispersed, ensuring a focus on maintaining cohesion:

$$\text{Distance Penalty} = -0.5 \cdot \text{Total Distance of Allies}$$

Random Bias

Adds a small random factor to break ties and introduce variability in decision-making:

$$\text{Random Bias} = \text{Uniform}(-0.1, 0.1)$$

Architecture :

Our implemention uses a **modular design** that separates core functionalities into distinct components.
This design ensures scalability, ease of maintenance, and reusability.
The main components include :

Game State Representation: GameMap Class

- Tracks positions and counts of humans, vampires, and wolves.
- Uses **dictionaries** for entities: {(x, y): count} for efficient lookups and updates.
- **Grid size** stored as a NumPy array: [height, width].
- Boolean flag is_vampire identifies the player's species (True for vampires, False for wolves).
- Unified Map State map_state : Combines the data from dictionaries into a single, structured list for a comprehensive snapshot of the game state.
  - Format: (x, y, num_humans, num_vampires, num_wolves).
  - Example :

```
map_state = [ (0, 0, 5, 0, 0), # 5 humans at (0, 0)
              (1, 1, 0, 3, 0), # 3 vampires at (1, 1)
              (2, 2, 0, 0, 4) # 4 wolves at (2, 2)]
```

Move Generation : calculate_possible_moves()

- Dynamically calculates valid moves for entities based on the current game state and game rules.
- Considers constraints like adjacency and available entity counts.

Architecture :

Our implemention uses a **modular design** that separates core functionalities into distinct components.
This design ensures scalability, ease of maintenance, and reusability.
The main components include :

Decision Tree Construction : Node and Tree Class

- Core Process:
  - Builds a decision tree up to a fixed depth of 3.
  - Alternates between Maximizing Layers (AI's turn) and Minimizing Layers (Opponent's turn).
  - Nodes store:
    - Current game state.
    - Possible moves.
    - Utility values evaluated by the heuristic function.
- Depth Selection:
  - Fixed at 3 to ensure:
    - Evaluation concludes on the AI's turn, reflecting an optimized position.
    - Avoids influence from the opponent's moves.
  - Balances computational efficiency and strategic depth.

Architecture :

Our implemention uses a **modular design** that separates core functionalities into distinct components.
This design ensures scalability, ease of maintenance, and reusability.
The main components include :

Heuristic Evaluation :

- **Formula :**

$$\text{Evaluation Score} = 10 \cdot (U_{\text{ai}} - U_{\text{op}}) + w_{\text{resource}} \cdot (\text{Resource Gain}_{\text{ai}} - \text{Resource Gain}_{\text{op}}) + w_{\text{attack}} \cdot (\text{Attack Power}_{\text{ai}}) + \text{Strategic Positioning} + \text{Distance Penalty} + \text{Random Bias}$$

- **Adaptation :**
  - Early Game: Focus on resource gain and attack power.
  - Late Game: Emphasizes strategic positioning and grid control.

Alpha-Beta Pruning

- Optimization Highlights:
  - Prunes branches where $\beta \leq \alpha$, reducing computational overhead.
  - Ensures real-time decision-making within the strict 2-second time limit.
- Benefits:
  - Focuses on the most promising moves.
  - Maintains efficiency without compromising accuracy.

Architecture :

Our implemention uses a **modular design** that separates core functionalities into distinct components.
This design ensures scalability, ease of maintenance, and reusability.
The main components include :

Server Communication :

- Maintains real-time synchronization with the server.
- Input: Updates from the server (opponent moves, battle results, etc.).
- Output: Sends the optimal move back to the server (e.g., source, target, units).

AI Decision-Making Process :

Input : Server provides grid, entity positions, and game updates.

Process :

Game State Update

- Synchronizes the AI's internal state with server data and updates grid and species information

Generate Moves

- Calculates possible moves for the AI based on the current game state and game rules.

Decision Tree

- Constructs a decision tree representing potential game states up to a predefined depth.
- Alternates between maximizing layers (AI's turn) and minimizing layers (opponent's turn).

Heuristic Evaluation

- Scores each potential game state in the decision tree based on metrics such as Resource gain, Attack potential, and Strategic positioning.

Alpha-Beta Pruning

- Prunes unnecessary branches of the decision tree to reduce computational overhead.
- Focuses only on the most promising moves.

Output :

- Send the selected move to the server, specifying source, target, and units.
- Resolve probabilistic outcomes if battles occur.

Iterative Improvement :

- Dynamically refines decision-making based on new updates from the server after each turn.
- Adjusts the game state representation and strategy for subsequent moves.
- Ensures adaptability to the evolving game scenario.

Testing and Results :

Testing :

Verified Capabilities :

- **Server Interaction :** Reliable connection and real-time processing of updates.
- **Move Generation :** All moves comply with game rules and adapt dynamically to the game state.

Pending Full Testing :

Comprehensive strategy testing will occur during the competition to evaluate performance against other AIs.

Observations :

Strengths:

- **Adaptability :** Heuristics adjust based on game state and priorities.
- **Efficiency :** Alpha-Beta pruning ensures moves are computed quickly.
- **Synchronization :** Accurate game state updates for real-time decisions.

Weaknesses:

- **Tree Depth Limitations :** Shallow search depth can miss long-term strategies.
- **Late-Game Complexity :** Sparse grids expose gaps in heuristic adaptation.
- **Uncertainty in Battles :** Probabilistic combat adds variability to outcomes.

Limitations, Next Steps and Conclusion :

Limitations

- **No Splitting Groups:** AI cannot split groups for parallel actions, reducing flexibility.
- **Computational Complexity:** Larger grids and deeper trees increase search space.
- **Simplified Heuristics:** Metrics do not fully account for multi-step strategies.
- **Randomness:** Probabilistic battles remain unpredictable in decision-making.

Next Steps

1. **Reinforcement Learning:**
   - Train AI via self-play to learn advanced strategies.
2. **Monte Carlo Tree Search (MCTS):**
   - Introduce stochastic methods for handling uncertainty.
3. **Scalability Improvements:**
   - Optimize performance for larger grids and deeper searches.

Summary :

- Demonstrates strong adherence to game rules and efficient decision-making.
- Highlights strengths in adaptability and real-time updates.
- Full performance evaluation to be conducted during the competition.

CentraleSupélec | université PARIS-SACLAY

MScAI
Master of Science in Artificial Intelligence
Applied to society

Thank You