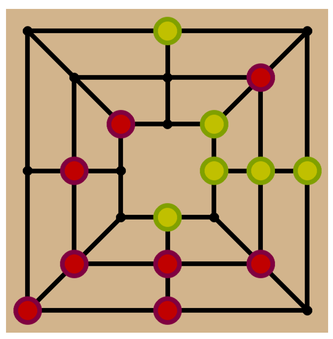
Intelligent Systems Seminar I(B): Nine Men's Morris with Minimax Algorithm

Marija Ćetković, Dmytro Tupkalenko, Diar Gashi

Introduction

This project focuses on designing and implementing an intelligent agent capable of playing **Nine Men's Morris (Mill)**. The agent's decisions are guided by the Minimax algorithm, a classical adversarial search method, which we integrated with the *famnit-gym* [2] environment for managing game states and generating legal moves. To enhance efficiency, the algorithm was supplemented with several strategic optimizations. A notable outcome is the implementation of multiple AI difficulty levels, allowing human players to engage with the agent through a responsive and interactive interface.

Game Setup



In our version of the game each player has 9 pieces and 24 available positions. The game is played in three phases:

- **Placement Phase:** Players alternately place their 9 pieces on empty board intersections (positions)
- **Movement Phase:** Once all pieces are placed, players slide pieces to empty adjacent positions
- **Flying Phase:** When reduced to 3 pieces, a player may "fly" to any empty position

Key Rules:

- Forming a **mill** (3 pieces in a row) allows capturing an opponent's piece
- Capturing opponent's piece that is part of a mill is not allowed, unless no other option is available
- The game ends as a lost, when a player who is to make a move, is reduced to 2 pieces or cannot make a legal move, it ends as a draw if total of 200 moves have been played.

Minimax Implementations

Basic Implementation → Alpha-Beta Pruning → Move Ordering → Hashing

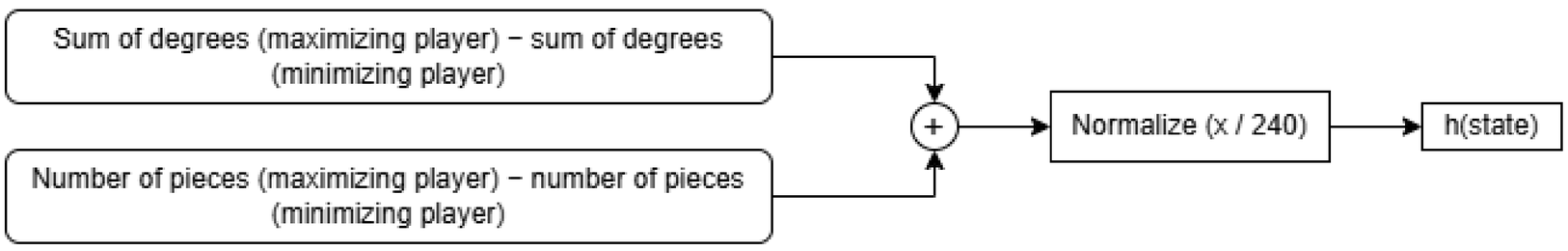
Algorithm	Computation Time
Basic Minimax	1019.9218 s
Alpha-Beta	5.4367 s
Alpha-Beta with Move Ordering	1.5387 s
Alpha-Beta with Move Ordering and Hashing	1.6504 s

Algorithm	Computation Time
Alpha-Beta with Move Ordering	35.6994 s
Alpha-Beta with Move Ordering and Hashing	35.9345 s

Evaluation Scenarios:

1. Time to find the optimal winning move for a predefined state when four moves from a draw.
2. Hashing vs. Non-hashing over 100 iterations when 7 moves from a draw.

Final Implementation: Because the full search space (depth is 200 nodes) is computationally infeasible, the search depth is limited. At maximum depth, non-terminal states are evaluated using the following heuristic function (the same one without normalization was used for move ordering):



Plot of maximum depth vs. computation time, total moves + win rate when AI (variable depth) plays against depth-1 AI with 10% random moves:

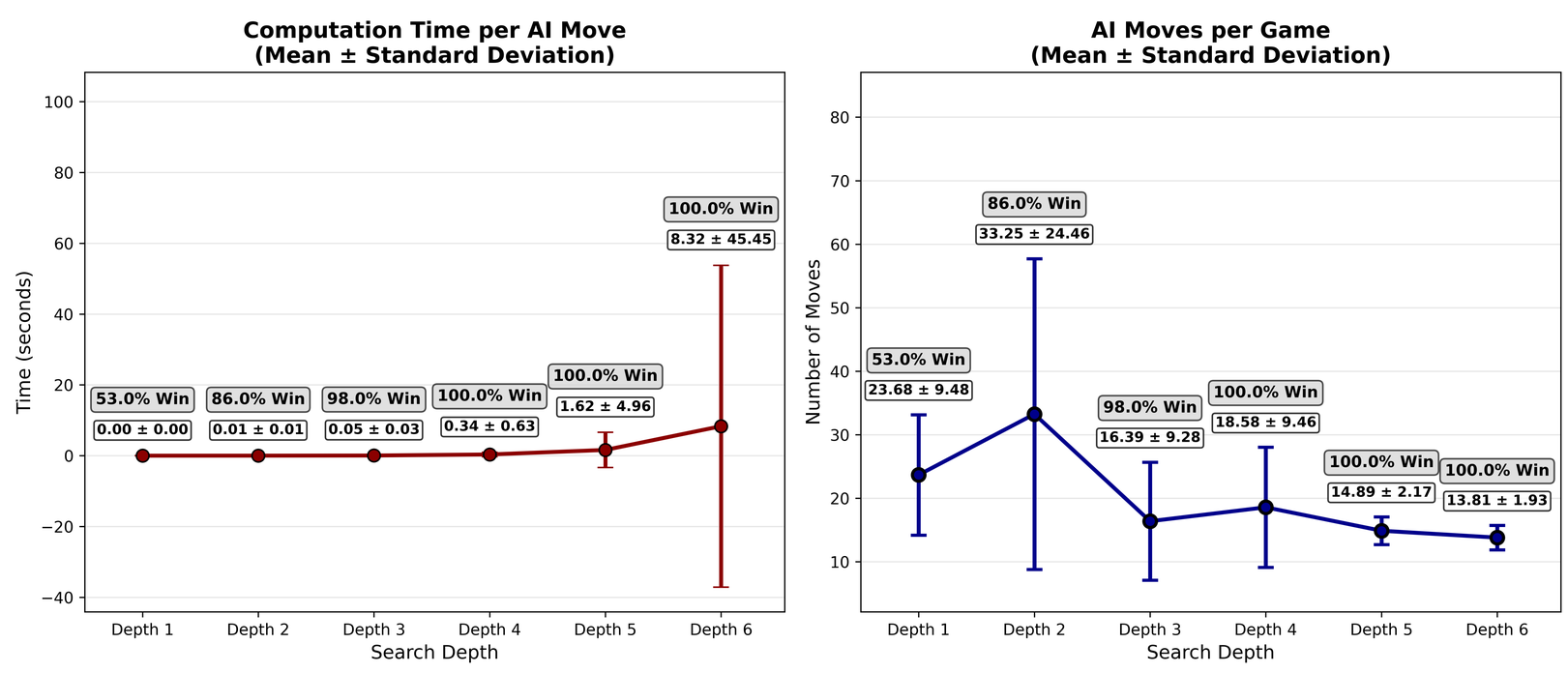


Table of game outcomes when an AI with limited depth plays against itself. Interestingly, on depth 6, the AI draws - which aligns with expectations, as a slightly different but similar version of the game has been proven to be a draw [1]. The conclusion we can draw from this is that this heuristic function is close to being "admissible" in the sense that: $\lim_{\text{max\_depth}\to 200} \text{heuristic} = \text{true value}$

Maximum Depth	Winner	Total Moves
1	Player 1	49
2	Player 1	55
3	Player 1	33
4	Player 1	39
5	Player 1	33
6	Draw	200

Difficulty Levels

- **Apprentice:** **100%** chance of a random move.
- **Adventurer:** max depth 2, **40%** chance of a random move.
- **Knight:** max depth 3, **30%** chance of a random move.
- **Champion:** max depth 5, **10%** chance of a random move.
- **Legend:** max depth 6, deterministic move selection (**0%** random moves).

Benchmarking: We ran a round-robin tournament among all difficulty levels. The overall performance shows a clear progression: higher-depth and less random AI consistently outperformed weaker ones:

Matchup	AI 1 Wins	AI 2 Wins
Apprentice vs Adventurer	0/10	10/10
Apprentice vs Knight	0/10	10/10
Apprentice vs Champion	0/10	10/10
Apprentice vs Legend	0/10	10/10
Adventurer vs Knight	0/10	10/10
Adventurer vs Champion	1/10	9/10
Adventurer vs Legend	0/10	10/10
Knight vs Champion	3/10	7/10
Knight vs Legend	1/10	8/10
Champion vs Legend	3/10	7/10

Difficulty	Wins	Win Rate	Draws	Losses
Apprentice	0/40	0.0%	0/40	40/40
Adventurer	11/40	27.5%	0/40	29/40
Knight	24/40	60.0%	1/40	15/40
Champion	29/40	72.5%	0/40	11/40
Legend	35/40	87.5%	1/40	4/40

Attempt to Prove Correctness

We attempted to prove that our minimax algorithm is correct and complete. To do this, we set up a predefined board from [1] known to be a draw (assuming both players play optimally) and checked whether the algorithm evaluates all moves as leading to a draw or considers any as potentially winning or losing. We reduced the search depth to 191, which remains computationally infeasible, so a full proof was not possible. The maximal depth explored was 12 (took 57 minutes), for which the algorithm guaranteed a draw - as expected - but this cannot constitute a complete proof, since at depth 13 it might evaluate differently.

References

1. R. Gasser, *Solving Nine Men's Morris*, Journal of Game Theory, 1996.
2. D. Šoberl, Famnit Gym environment: https://github.com/DomenSoberlFamnit/famnit-gym
3. Project source code: https://github.com/marijacetkovic/mill_project