

# PLAYING NINE MEN'S MORRIS: AN AI AGENT USING ADVERSARIAL SEARCH

Nine Men's Morris (Mill) is a classic two-player strategy game divided into three phases: placing, moving, and flying. The goal is to form mills three aligned pieces to capture the opponent's pieces. This project explores the design of an intelligent AI agent that plays Nine Men's Morris using adversarial search.



## Author

Dushime Mudahera Richard  
Masters in Data Science  
Faculty of Mathematics, Natural Sciences, and Information Technologies  
University of Primorska (Famnit)

## INTRODUCTION

- Nine Men's Morris: classic two-player strategy board game
- Three phases: placing, moving, flying
- Objective: Create 'mills' (three in-a-row) to capture opponent pieces

## METHODOLOGY

### Project Objectives

- Design intelligent game agent
- Ensure real-time playability
- Analyze strategic strength

### Utility Function

- Piece Count, Mills, Mobility, Threats, Phase Bonus
- Final Score: Weighted sum (details in visual box)

## AIM

- Develop an AI agent using adversarial search
- Implement multiple difficulty levels
- Evaluate performance and playability

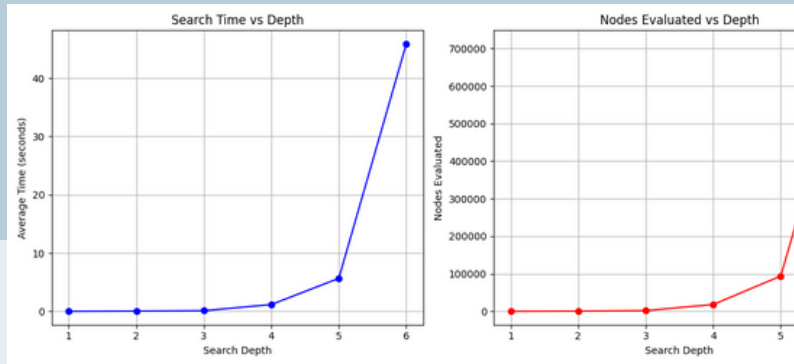
## RESULTS

### Depth Analysis

- Search depth beyond 6 = infeasible for real-time
- Alpha-beta pruning: ~60-80% reduction in explored nodes
- Optimal depth: 5-6 for <5s move time

### Key Graphs:

- Search Time vs Depth
- Nodes Evaluated vs Depth



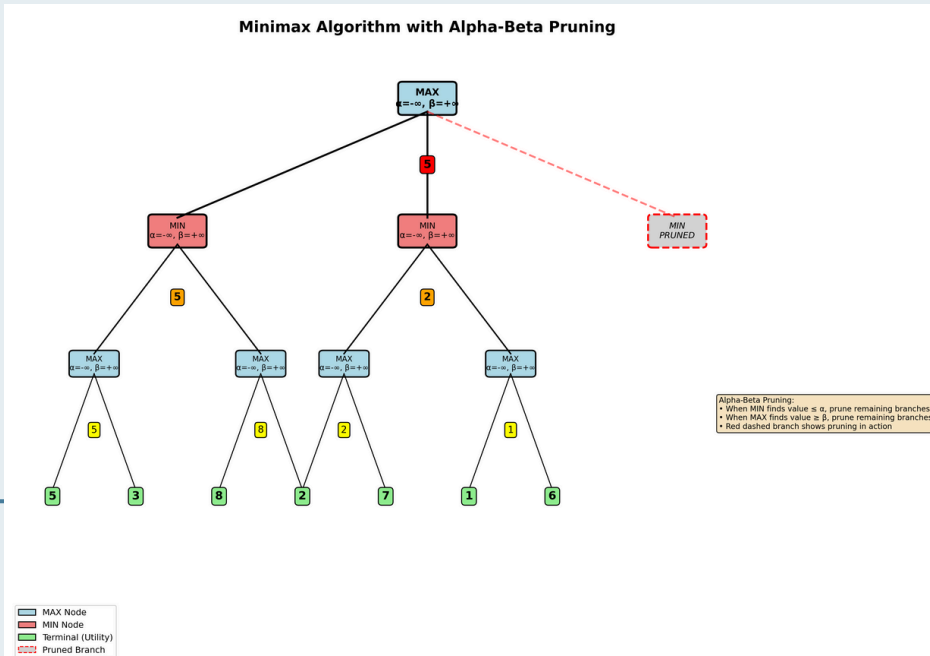
## IMPLEMENTATION

### Features

- Custom minimax & alpha-beta implementations
- Adaptable UI
- Tournament mode
- Code in Python; famnit-gym for environment

## DIFFICULT LEVELS

Level: Easy	Medium	Hard
Depth: 2	4	6
Random: 30%	15%	0%
Desc: Randomized moves	Occasional errors	Full adversarial



## CONCLUSION

- Alpha-beta pruning makes the AI fast enough for real-time play.
- Our utility function creates strong and adaptable gameplay.
- Difficulty levels ensure the game is challenging for all players.
- When two strong AIs play, most games end in a draw, showing solid strategy.

### Github Repository

For More Visit the Repository [here](#)

Utility Function Components		
Utility = Piece Score + Mill Score + Mobility Score + Phase Bonus + Threat Score		
All scores evaluated from Player 1 perspective (higher = better)		
Piece Count	(my_pieces - opp_pieces) * 10 Difference in pieces on board	Weight: 10.0
Mill Count	(my_mills - opp_mills) * 30 Number of completed mills (3 in a row)	Weight: 30.0 (highest)
Mobility	(my_moves - opp_moves) * 2 Number of legal moves available	Weight: 2.0
Phase Bonus	+5 if in placing phase Early game advantage	Weight: 5.0
Threat Detection	(my_threats - opp_threats) * 10 Potential mills (2 pieces, 1 empty)	Weight: 10.0
Example Calculation: Piece: (7-6) * 10 = +10   Mill: (1-0) * 30 = +30   Mob: (5-4) * 2 = +2   Phase: +5   Threat: (2-1) * 10 = +10 Total Utility = 10 + 30 + 2 + 5 + 10 = +57		

