4Bits presents

An overview of the best gaming bot: Tablut AI Agent

Francesco Giuseppe Ielo - 0001226296
Gabriele Napoletano - 0001222954
Simone Magli - 0001222264
Davide Giannetti - 0001223777

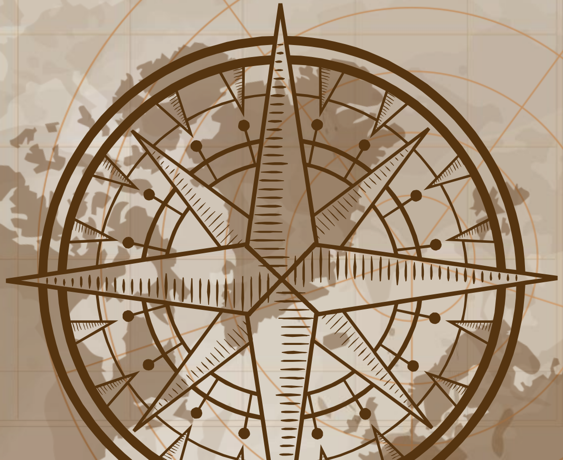


Table of contents

01

Introduction

Overview of the Tablut challenge, the game rules, and our agent's goals.

02

AI architecture

Core search logic using Minimax, Alpha-Beta pruning, and performance optimizations.

03

Heuristic

Evaluating game states based on King safety, material, and strategic positioning.

04

Conclusions

Summary of competition results and analysis of the bot's strengths.

05

Future developments

Proposed enhancements and next steps to further improve the AI.

01

Introduction

Tablut challenge, game rules and our agent's goals.

Rules of the game

Tablut is an asymmetric strategy game played on a 9x9 board, involving two opposing sides:

1. Factions and Goals:

- **White (Defender):** Controls the King (K) and 8 pawns. Goal: Move the King to any of the four corner squares.
- **Black (Attacker):** Controls 16 pawns. Goal: Capture the King.

2. Basic Movement:

- All pieces move any number of free squares horizontally or vertically (Rook movement in Chess).
- Diagonal movement is not allowed.

3. Board Restrictions:

- **Throne (Center):** Only the King can enter or leave this square. No piece may pass over it.
- **Citadels (Marked Squares):** Pawns cannot land on these squares.

4. Captures:

- A common piece (White or Black) is captured and removed when flanked orthogonally on two opposite sides by two opponent pieces, the Throne, or a Citadel.

5. Win/Loss Conditions:

- **White Wins:** The King reaches an outer corner square.
- **Black Wins:** The King is surrounded and captured. Capture conditions vary (2, 3, or 4 sides) depending on the King's proximity to the Throne or Citadels.
- **Draw:** Declared after a time limit, repeated board positions, or excessive non-capturing moves.

Objective: Building a High-Performance Tablut AI

Core AI Goals

- **Implement a robust Search Algorithm:** Develop an optimized **Minimax search** with **Alpha-Beta Pruning** capable of achieving high search depth.
- **Maximize Performance:** Utilize advanced techniques to ensure decision-making within the server's time limits (e.g., **Iterative Deepening** and **Root Node Parallelization**).
- **Ensure Correctness:** Implement precise Ashton Tablut rules and game logic to ensure all generated moves are legal.

Technical & Optimization Objectives

- **Efficient State Representation:** Employ an optimized game state (*FastTablutState*) that incorporates **Zobrist Hashing** for **Transposition Table** lookup, drastically accelerating the search.
- **Solve the Horizon Effect:** Integrate a **Quiescence Search** mechanism to evaluate tactical situations (captures/threats) accurately at the search horizon.

02

AI architecture

Minimax algorithm, alpha-beta pruning technique.

Minimax Algorithm

A foundational recursive search algorithm in Game Theory.
It is designed for two-player, **zero-sum games** where the players alternate turns

Mechanism:

It explores the game tree to find the move that provides the best outcome for the agent, assuming the opponent plays optimally to minimize the agent's advantage.

MAX (white player)

Attempts to maximize the utility value.

MIN (black player)

Attempts to minimize the maximum gain achievable by MAX.

We used this algorithm for Tablut because it is a perfect-information, deterministic game suitable for exhaustive search. It guarantees the most rational move up to the search depth, providing a strong strategic foundation for competitive play.

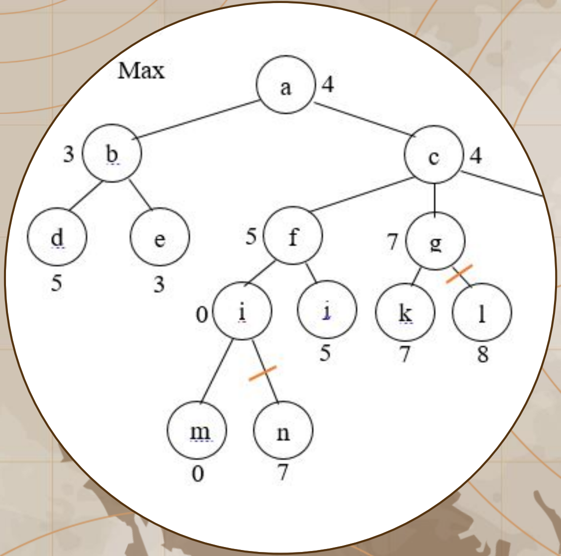
Alpha-Beta Pruning: cutting useless branches

- An extension of the Minimax algorithm that aims to reduce the nodes explored in the game tree.
- Result: An exponential increase in the search depth achieved within the same time limit.

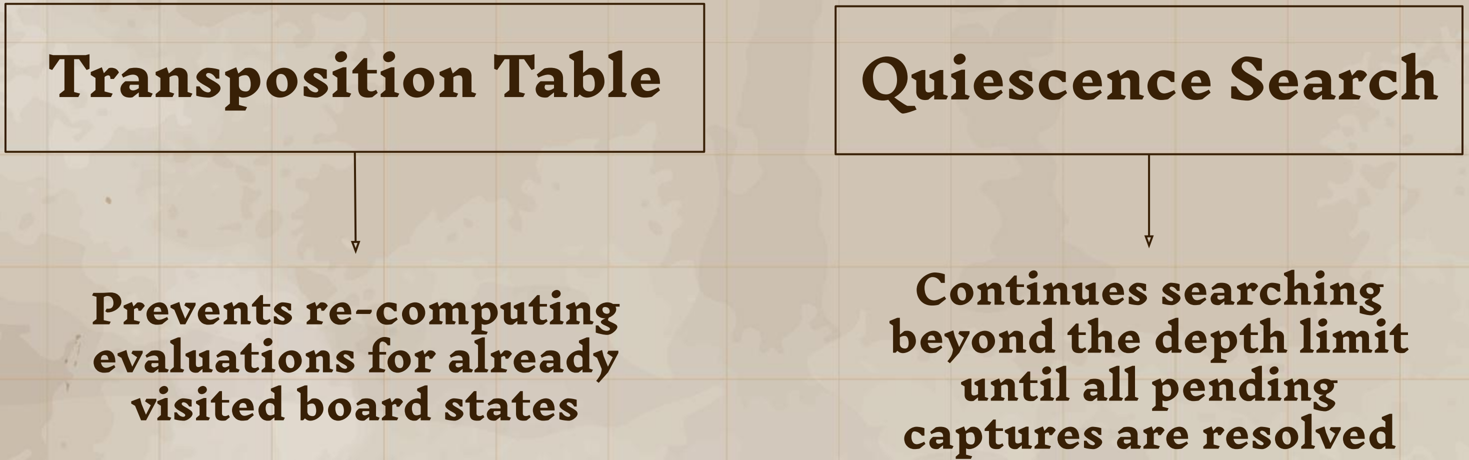
Pruning occurs when we find a potential move that is worse than an alternative already secured.
If a MIN node (black) guarantees a lower score (β) than the maximum score (α) that MAX (white) has already secured, the opponent will never choose that branch, so it is ignored (pruned).

Tablut Agent Advantage:

- Crucial for meeting the server's time constraint (1 minute per move).
- Allows the bot to explore a significantly wider game space, drastically improving the strategic quality of its final decisions.



Implementation and Optimization



Transposition Table

Prevents re-computing evaluations for already visited board states

Implemented with Zobrist Hashing

Quiescence Search

Continues searching beyond the depth limit until all pending captures are resolved

Limited to a maximum depth of 2

03

Heuristic

Evaluating the game by establishing different weights for different positions.

The Heuristic Function

Valuing the **Board State**: transforming a position into a numerical score to guide the Alpha-Beta search.

Material Advantage

- Material balance on the board
- Difference between the number of Black pawns and White pawns.

King Safety and Position

- Evaluation of the King's immediate safety (e.g., surrounding Black pieces)
- Penalties if the King is adjacent to the Throne or other restrictive squares.

Escape Route Potential

- The King is awarded a score based on its Manhattan distance to the nearest empty escape square (corner).
- Points for available, unblocked "escape paths".

04

Conclusions

Looking at the general results.

Conclusions

Time Management

The agent maximizes search depth by effectively utilizing the entire available turn time via Iterative Deepening, ensuring high-precision moves without risking timeouts.

Winning Capability

By balancing material advantage with King safety heuristics, the bot demonstrates the ability to convert tactical opportunities into actual victories, proving effectiveness in match scenarios.

05

Future developments

Possible enhancements and steps.

Future developments

Opening book

Implementation of a pre-calculated opening database to play early turns instantly, saving search time and steering the game toward favorable configurations.

Endgame tablebases

Creation of a database of solved endgames to ensure perfect play in the final stages (with few pieces left) without relying on the search tree.

Genetic algorithm

Intensive execution of the genetic engine to automatically refine heuristic weights via self-play, overcoming manual tuning limitations.

Thanks!

Francesco Giuseppe Ielo - 0001226296
Gabriele Napoletano - 0001222954
Simone Magli - 0001222264
Davide Giannetti - 0001223777