Chess with Enhanced AI and Strategic Variations

Syed Hussamuddin (22K-4658)
Usman Sohail (22K-4487)
Syed Muqeet ur Rehman (22K-4477)

15th May 2025

Course: Artificial Intelligence
Instructor Theory: Miss Alina Arshad
Instructor Lab: Miss Syeda Ravia Ejaz

Abstract

This report presents the design and implementation of a chess variant with enhanced artificial intelligence. The game, developed using Python and Pygame, introduces a two-queen configuration for each player and integrates both the Minimax algorithm and Reinforcement Learning techniques to create a challenging AI opponent. This project explores the implications of this variation on gameplay dynamics and details the AI strategies used.

1. Project Overview

1.1. Project Topic

The project involves the development of a chess game with a custom variant where each player starts with two queens instead of one. The main focus lies in the creation of an intelligent AI opponent using the Minimax algorithm with Alpha-Beta pruning, complemented by a secondary AI model trained via Reinforcement Learning.

1.2. Objective

The primary goal is to construct a functional and strategic chess engine capable of handling complex game states introduced by the two-queen variant. The AI should be able to evaluate positions and make optimal decisions under both deterministic (Minimax) and probabilistic (RL) paradigms.

2. Game Description

2.1. Original Game Background

Chess is a well-known two-player strategy game played on an 8x8 board. Each player traditionally starts with 16 pieces. The game ends when a player's king is checkmated.

2.2. Innovations Introduced

- **Two Queens Variation**: Each player begins with two queens, one replacing a pawn.
- **Modified Strategy Requirements**: The AI must accommodate more powerful early-game moves due to additional queens.
- **Advanced AI Integration**: We use Minimax with Alpha-Beta pruning and Reinforcement Learning for AI-based decision making.
- **Customizable Difficulty**: Players can adjust AI depth or training-based behavior to alter difficulty.

3. AI Approach and Methodology

3.1. Minimax Algorithm with Alpha-Beta Pruning

The core AI decision-making algorithm is Minimax, used to simulate possible future game states and determine optimal moves. Alpha-Beta pruning reduces unnecessary computation by eliminating branches that cannot affect the final decision.

3.2. Reinforcement Learning

A secondary AI model is trained using self-play and reward-based feedback. The AI learns to associate game states with high-reward moves over time, providing a dynamic and adaptive opponent.

3.3. Complexity Analysis

Minimax alone has a time complexity of $O(b^d)$, where b is the branching factor and d is depth. With Alpha-Beta pruning, this improves to approximately $O(b^{d/2})$. However, the introduction of a second queen increases b, requiring optimizations to maintain real-time performance.

4. Game Rules and Mechanics

4.1. Modified Rules

- Each player starts with two queens.
- The second queen replaces one of the pawns (usually the d-pawn).
- All standard rules, including castling, en passant, and promotion, are preserved.

4.2. Winning Conditions

- Standard checkmate rules apply.
- Draw and stalemate rules remain unchanged.

4.3. Turn Sequence

Players alternate turns, and the AI moves immediately after the human player. AI decisions are based on the current mode (Minimax or RL).

5. Implementation Plan

5.1. Language and Tools

- **Language**: Python
- **Libraries**: Pygame (GUI), NumPy (data handling), custom AI modules

5.2. Milestones

1. **Weeks 1-2**: Setup game board, piece movement, and rendering.
2. **Weeks 3-4**: Implement Minimax with Alpha-Beta pruning.
3. **Weeks 5-6**: Introduce two-queen variant logic and validate gameplay.
4. **Week 7**: Integrate RL model and user interface enhancements.
5. **Week 8**: Final testing, debugging, and documentation.

6. Results and Evaluation

The Minimax AI with Alpha-Beta pruning shows competent move selection even with the increased complexity from two queens. The reinforcement learning agent displays adaptive behavior over time but requires extensive training for consistent high-level play. Combining both approaches creates a versatile and challenging AI.

7. Conclusion

This project successfully implements a variant of chess that increases strategic depth and integrates two distinct AI approaches. The use of both Minimax and Reinforcement Learning provides insight into AI decision-making and demonstrates how different strategies perform under modified rules.

8. References

- Python Pygame Documentation
- AI in Chess: Minimax and Alpha-Beta Pruning Tutorials
- Research Papers on Reinforcement Learning in Board Games
- YouTube Series: Chess AI with Python