Artificial Intelligence

Report

"Mini-Checkers with Obstacles"

NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES

KARACHI CAMPUS

Group Members:

Muhammad Muzammil	**22K-4267**
Asim Majeed	**22K-4535**
Ayan Hasan	**22K-4367**

Instructor : Talha Shahid

1. Executive Summary

- Project Overview:
  - This project introduces a modified version of the classic checkers game by implementing a 6×6 board with dynamic obstacles that alter gameplay strategies. The main objective was to design and develop a Minimax-based AI agent enhanced with alpha-beta pruning to compete against a human player. The obstacle-aware heuristic was engineered to evaluate game states based on piece material, movement mobility, and path clarity.

2. Introduction

- Background:
  - Checkers is a traditional two-player strategy board game involving diagonal movement and jumping over opponent pieces to capture them. The decision to enhance checkers came from the desire to increase complexity and strategic depth by introducing impassable obstacles. This added a layer of unpredictability and required a more intelligent AI.

- Objectives of the Project:
  - Develop a dynamic obstacle system that changes mid-game.
  - Implement an AI opponent using Minimax with alpha-beta pruning.
  - Design an obstacle-aware heuristic to evaluate moves.
  - Build a user-friendly GUI with Pygame.
  - Provide a console-based alternative for gameplay.

3. Game Description

- Original Game Rules:
  - In traditional checkers, players take turns moving their pieces diagonally on dark squares. Capturing is performed by jumping over an opponent's piece. The objective is to capture all opponent pieces or block them from moving.

- Innovations and Modifications:
  - Board size reduced to 6×6.
  - Randomly generated obstacle squares that block movement and capture.
  - Mid-game obstacle re-randomization.
  - Obstacle-aware AI decision-making.

4. AI Approach and Methodology

- AI Techniques Used:
  - We used the Minimax algorithm with alpha-beta pruning to allow the AI to efficiently search possible game states. Reinforcement learning was also considered for tuning heuristic weights via self-play.

- Algorithm and Heuristic Design:
  - The heuristic evaluates a board state by scoring material advantage, mobility (number of available legal moves), and path clarity (accessibility of unblocked routes).

- AI Performance Evaluation:
  - AI performance was evaluated through multiple playthroughs against human players, measuring win rates, decision-making speed, and quality of moves.

5. Game Mechanics and Rules

- Modified Game Rules:
  - Players cannot move into or jump over obstacle squares.
  - Obstacles are re-randomizable mid-game to introduce dynamic challenges.
  - Game ends when one player has no legal moves or no remaining pieces.
- Turn-based Mechanics:
  - Players alternate turns. On each turn, a player selects a piece and chooses a legal move. The AI responds with its best move based on Minimax search.
- Winning Conditions:
  - The player who captures all opponent pieces or blocks all legal moves wins the game.

6. Implementation and Development

- Development Process:
  - The project was developed using Python and the Pygame library. Game rules and board logic were encapsulated in Python modules. AI logic using Minimax with pruning was developed and integrated into the gameplay loop.

- Programming Languages and Tools:
  - Programming Language: Python
  - Libraries: Pygame, NumPy
  - Tools: GitHub for version control

- Challenges Encountered:
  - Handling moves legality in the presence of obstacles.
  - Tuning the heuristic to balance between aggression and mobility.
  - Designing a responsive and intuitive UI with Pygame.

7. *Team Contributions*

- Team Members and Responsibilities:
  - Saiyed Asim Majeed: Responsible for AI algorithm development (Minimax, Alpha-Beta Pruning).
  - Muhammad Muzammil: Handled game rule modifications and board design.
  - Ayan Hasan: Focused on implementing the user interface and integrating AI with gameplay.

8. *Results and Discussion*

- AI Performance:
  - The AI achieved a win rate of approximately 75% against novice human players. The average decision-making time was under 1 seconds per move. Obstacle-awareness significantly enhanced the AI's ability to avoid traps and capitalize on positional advantages.

9. *References*

- Russell, S., & Norvig, P. (2021). Artificial Intelligence: A Modern Approach.

- Pygame documentation: https://www.pygame.org/docs/

- Checkers game rules: https://en.wikipedia.org/wiki/Draughts

- Alpha-beta pruning: https://en.wikipedia.org/wiki/Alpha%E2%80%93beta_pruning