National University of Computer & Emerging Sciences

Karachi Campus

TITLE OF PROJECT

Advanced Checkers Game with Intelligent AI Opponents

Project Proposal

Artificial Intelligence

Section: B

Group Members:

- **22k-4282 Daniyal Fahim**
- **22k-4167 Daiyan Ur Rehman**
- **22k-4663 Dawood adnan**

Project Proposal

Introduction

The proposed system is an **Advanced Checkers Game** developed as a desktop application, designed to provide an interactive and educational platform for exploring artificial intelligence techniques. The game implements the classic board game of Checkers (Draughts) with a user-friendly graphical interface built using Python's Tkinter library. It features three distinct AI algorithms—Minimax, Negamax, and Monte Carlo Tree Search (MCTS)—to serve as intelligent opponents for the human player. Players can customize their experience by selecting their name, AI difficulty (Easy, Medium, Hard), and AI algorithm. The system also includes features like move history tracking, responsive board resizing, and a modern visual design to enhance user engagement. This project aims to demonstrate the application of AI in game development, showcasing how different algorithms perform in a strategic board game context.

Existing System

Several Checkers games with AI opponents exist, ranging from web-based applications to mobile and desktop games. Notable examples include:

- **Online Checkers Games (e.g., 247 Checkers, Coolmath Games)**: These are web-based platforms offering basic Checkers gameplay against rule-based AI opponents. They typically use simple heuristics or pre-programmed strategies, lacking advanced AI algorithms like Minimax or MCTS. The user interface is often minimalistic, with limited customization options.
- **Open-Source Checkers Projects (e.g., GitHub Repositories)**: Some open-source implementations use Python or JavaScript with basic AI, often employing Minimax with alpha-beta pruning. However, these projects rarely offer multiple AI algorithms or difficulty levels, and their user interfaces are not always polished.
- **Commercial Checkers Software (e.g., Checkers Deluxe)**: These provide polished graphics and AI opponents but are closed-source, limiting educational value. The AI logic is often proprietary, and customization options are restricted to basic settings like difficulty.

While these systems provide functional Checkers gameplay, they often lack a combination of advanced AI techniques, user customization, and educational transparency, which are critical for an AI-focused academic project.

Problem Statement

The existing Checkers systems have several limitations:

1. **Limited AI Variety**: Most systems rely on a single AI algorithm (e.g., basic heuristics or Minimax), limiting the ability to compare different AI approaches in a single application.
2. **Lack of Customization**: Few systems allow users to adjust AI difficulty or select specific algorithms, reducing flexibility for both casual players and those studying AI.
3. **Basic User Interfaces**: Many open-source or free Checkers games have outdated or minimalistic interfaces, lacking modern design elements like responsive layouts or visual feedback for valid moves.
4. **Limited Educational Value**: Existing systems often do not expose AI logic or provide insights into how different algorithms perform, making them less suitable for academic purposes.
5. **No Move History Tracking**: Most systems do not record game moves, which is useful for analyzing gameplay or learning from past matches.

To address these gaps, our system introduces multiple AI algorithms, customizable settings, a modern GUI, and educational features like move history and algorithm performance insights.

Proposed Solution

The proposed Advanced Checkers Game addresses the identified problems by incorporating the following solutions:

1. **Multiple AI Algorithms**: Implement three AI algorithms—Minimax with alpha-beta pruning, Negamax with alpha-beta pruning, and Monte Carlo Tree Search (MCTS)—allowing users to select and compare their performance. This enables a deeper understanding of AI techniques in game playing.
2. **Customizable Gameplay**: Provide options to set player names, choose AI difficulty (Easy: depth 2, Medium: depth 3, Hard: depth 5 for Minimax/Negamax; 400/800 iterations for MCTS), and select the AI algorithm, catering to both casual and advanced users.
3. **Modern and Responsive GUI**: Develop a Tkinter-based interface with a visually appealing design, including a responsive board that adjusts to window size, highlighted valid moves, and a gradient-style piece rendering for aesthetic appeal.
4. **Educational Features**: Include a move history panel to track and review moves, and print AI move computation times to the console for performance analysis, making the system valuable for AI education.
5. **Optimized Performance**: Optimize AI algorithms (e.g., alpha-beta pruning in Minimax/Negamax, balanced MCTS iterations) to ensure smooth gameplay even on modest hardware, addressing performance issues in some existing systems.

These enhancements make the system a robust platform for both entertainment and learning AI concepts.

Salient Features

- **Interactive Checkers Gameplay**: Play Checkers against AI opponents with support for standard rules, including captures, king promotions, and mandatory jumps.
- **Three AI Algorithms**:
    - Minimax with alpha-beta pruning for deterministic move evaluation.
    - Negamax with alpha-beta pruning for a simplified yet efficient AI implementation.
    - Monte Carlo Tree Search for probabilistic, simulation-based decision-making.
- **Customizable Settings**: Select player name, AI difficulty (Easy, Medium, Hard), and AI algorithm (Minimax, Negamax, MCTS).
- **Responsive GUI**: Tkinter-based interface with a resizable board, modern color scheme, highlighted valid moves, and king piece indicators.
- **Move History Tracking**: Record and display all moves in a separate window for gameplay analysis.
- **Game Over Detection**: Automatically detect win/loss conditions (no pieces or no valid moves) with informative pop-up messages.
- **Performance Insights**: Display AI move computation times in the console for educational purposes.
- **Cross-Platform Compatibility**: Run on Windows, Linux, or macOS with minimal dependencies.

Tools & Technologies

- **Programming Language**: Python 3
- **Framework/Libraries**:
    - **Tkinter**: For building the graphical user interface.
    - **NumPy**: For efficient board representation and manipulation.
- **Operating System**: Cross-platform (Windows, Linux, macOS).
- **Development Tools**:
    - **VS Code** or **PyCharm** for coding and debugging.
    - **Git** for version control.
- **Dependencies**: Standard Python libraries (copy, random, time) and NumPy for array operations.