AI Project Report

22K-4609

22K-4202

Miss Alishba Subhani

Project Overview

This project involved modifying a Risk-inspired strategy game to support AI players and introduce random game events. The main goal was to make the game more dynamic and intelligent. Random events were added to affect territories and player advantages during gameplay. An AI player was implemented using alpha-beta pruning with a utility function based on the number of territories gained or lost and the loss given to the defender. This helped the AI make better decisions by looking a few moves ahead. The game can also be extended for training AI through reinforcement learning.

Introduction

- **Project Background:**

  The original game was a simple turn-based strategy game inspired by Risk, where players manually attacked and defended territories without any AI or dynamic elements. All decisions were made by human players, and the gameplay followed a fixed, predictable pattern with no random events or intelligent behavior.

- **Project Objectives:**

  The main objective of this project is to enhance the original strategy game by introducing random in-game events that can affect gameplay and incorporating an AI player that makes intelligent decisions using game theory techniques like alpha-beta pruning.

Game Description

- **Original Game Rules:**
  The original game is inspired by Risk, where players take turns to attack and defend territories on a world map. Each player starts with a certain number of territories and armies. The goal is to conquer all enemy territories through strategic attacks, reinforcements, and defense.

- **Innovations and Modifications:**
  This version introduces random game events that can influence gameplay, such as territory bonuses or penalties. Additionally, an AI player is implemented using alpha-beta pruning to make decisions, and a utility function is used to evaluate game states based on territorial advantage. These modifications aim to add unpredictability and intelligent decision-making to the game.

AI Approach and Methodology

- **AI Techniques Used**
  We used the Minimax algorithm enhanced with Alpha-Beta Pruning to enable the AI player to simulate and evaluate possible game states up to a limited depth. This allowed the AI to choose optimal moves based on the current game scenario in a competitive environment involving random events.

- **Algorithm and Heuristic Design**
  The AI uses a utility function that evaluates game states based on the change in the number of territories controlled from the root state. The more territories gained (and fewer lost) and the loss given to defender, the higher the utility score. This heuristic guides the AI in selecting moves that maximize territorial advantage while minimizing potential losses.

- **AI Performance Evaluation**
  The performance was assessed by tracking its win rate across multiple simulated games, its average decision-making time per move, and its ability to adapt to random game events. These metrics provided insight into the effectiveness and efficiency of the AI logic. At first, AI was consuming much time with unlimited depth. So we limit the depth to 3 stages after some experiments.

Game Mechanics and Rules

- **Modified Game Rules:**
  - Random events are introduced, which can alter the state of the game by granting or removing territories, affecting player positions, or triggering special conditions.
  - Players can gain or lose advantages based on these events, adding unpredictability to the strategy.
  - AI-controlled players follow the same rules as humans but use game-theory logic to make decisions.
- **Turn-based Mechanics:**
  - The game proceeds in turns, where each player takes action during their respective turn.
  - A player can choose to attack, defend, or respond to a random event.
  - After all players take their turns, a new round begins.
  - The game ends when one player dominates all territories or a certain game state threshold is reached.
- **Winning Conditions:**
  - A player wins by gaining control of all territories on the map.
  - Alternatively, if a predefined maximum number of turns is reached, the player controlling the most territories wins.

Implementation and Development

- **Development Process:**
  The game was developed using Python and the Pygame library to handle graphics and user interaction. The original game mechanics were expanded to include random events and AI players. The AI logic was implemented using the Minimax algorithm with Alpha-Beta pruning, allowing the AI to simulate possible future game states and make strategic decisions based on a heuristic evaluation function.

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

- **Challenges Encountered:**
  One major challenge was that after simulation, the AI player was not playing its turn correctly. It took a long debugging session to resolve this. After fixing the logic, the AI turn was not being visually updated, causing confusion during gameplay. This was addressed by forcing screen updates during AI turns. However, if a user interacts (clicks) during the AI's turn, the window sometimes becomes unresponsive, which remains a known issue.

Team Contributions:

- **Rao Abdul Hadi**: Implemented Game mechanics and GUI
- **Muhammad Taha Khan**: Optimized the GUI. Removed bugs and rendering issues
- **Muhammad Taha Khan**: Experiment and implement the AI algorithm development
- **Rao Abdul Hadi**: Modified the heuristic mechanism. Evaluated the AI player

Results and Discussions

AI performance: The AI player was able to win every 6/10th game. With limited depth, it utilizes only 2-3 seconds to take its decisions with help of heuristic evaluation on reaching maximum depth.

References

- Pygame Library
- Adverserial Search Lab Manual