Learning How to Play RISK with AI

Stephane Remigereau, Mohammadhossein Shafinia, Sandy Lee
TA: Will Deaderick, Stanford University

INTRODUCTION

- Risk is a strategic board game for 2 to 6 players with a simple objective: to conquer the world by occupying every territory on the board, thus eliminating all your opponents
- Our goal is to implement a game-playing agent for a simplified version (two-player zero sum game) of the board game Risk
- Motivation: Risk is an infinite state game and the outcome of the moves is non deterministic which makes it challenging to create an AI that can play smarter than a human

MODELING

- **Players:** Agent, Opponent
- **State:** (Player, {territory: (number of troops, controlling player)} for each territory *i,* Action type)
- **Actions:**
  Reinforce: player positions new troops in territories the player controls
  Attack: invades a neighboring territory occupied by enemy, with the battle conducted by rolling six-sided dice; can launch as many attacks as long as player has enough troops
  Then decides to end his turn
- **IsEnd:** If one player controls all the countries
- **Utility:** +Infinity if player wins, -Infinity if opponent wins

APPROACH

Agents

Human
Agent played by human, defined as our oracle

Random
Randomly chooses the reinforced territory, and randomly chooses the territory to attack with and the territory to invade

Aggressive
Reinforce the country with the most number of troops. Attack with the territory with the most number of troops, the opponent territory with the least number of troops, until it cannot attack

Semi-Smart Random
Random reinforcement. Random attack + attack until the number of dice for attack is less than or equal to the number of dice for defender

Semi-Smart Aggressive
Aggressive + attack until the number of dice for attack is less than or equal to the number of dice for defender

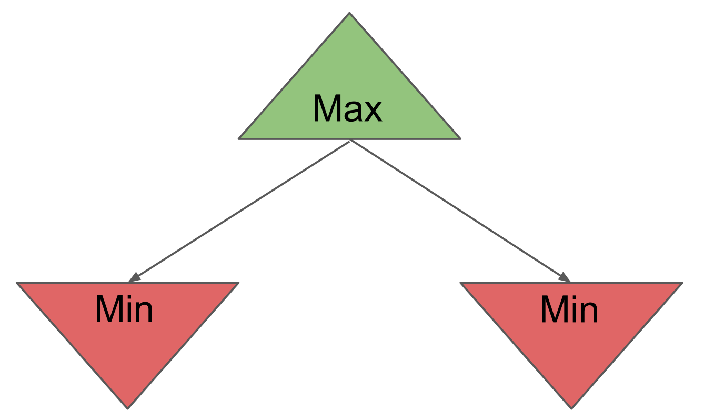
Expectiminimax
Performs the expectiminimax algorithm using an evaluation function and depth one

Evaluation Function for Expectiminimax

- For a territory, we define some features using domain knowledge:
  Border Security Threat (BST) = *Σ(# of neighboring enemy troops)*
  Border Security Ratio (BSR) = *BST/# of troops in the territory*
- **Evaluation:** *Σ(1/BSR of each country) + (# enemy territories/total # of territories)*

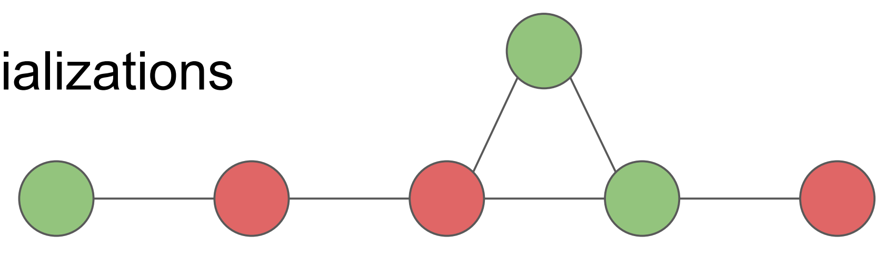
$$V_{expectiminimax}(s,d) = \begin{cases} Utility(s) & IsEnd(s) \\ Eval(s) & d=0 \\ \min_{a \in Actions(s)} \sum_{s'} \pi(s,a,s') V_{expectiminimax}(s',d) & Player = Agent \\ \max_{a \in Actions(s)} \sum_{s'} \pi(s,a,s') V_{expectiminimax}(s',d-1) & Player = Opponent \end{cases}$$

- Uses stochastic policies
  (the odds of various dice combinations)

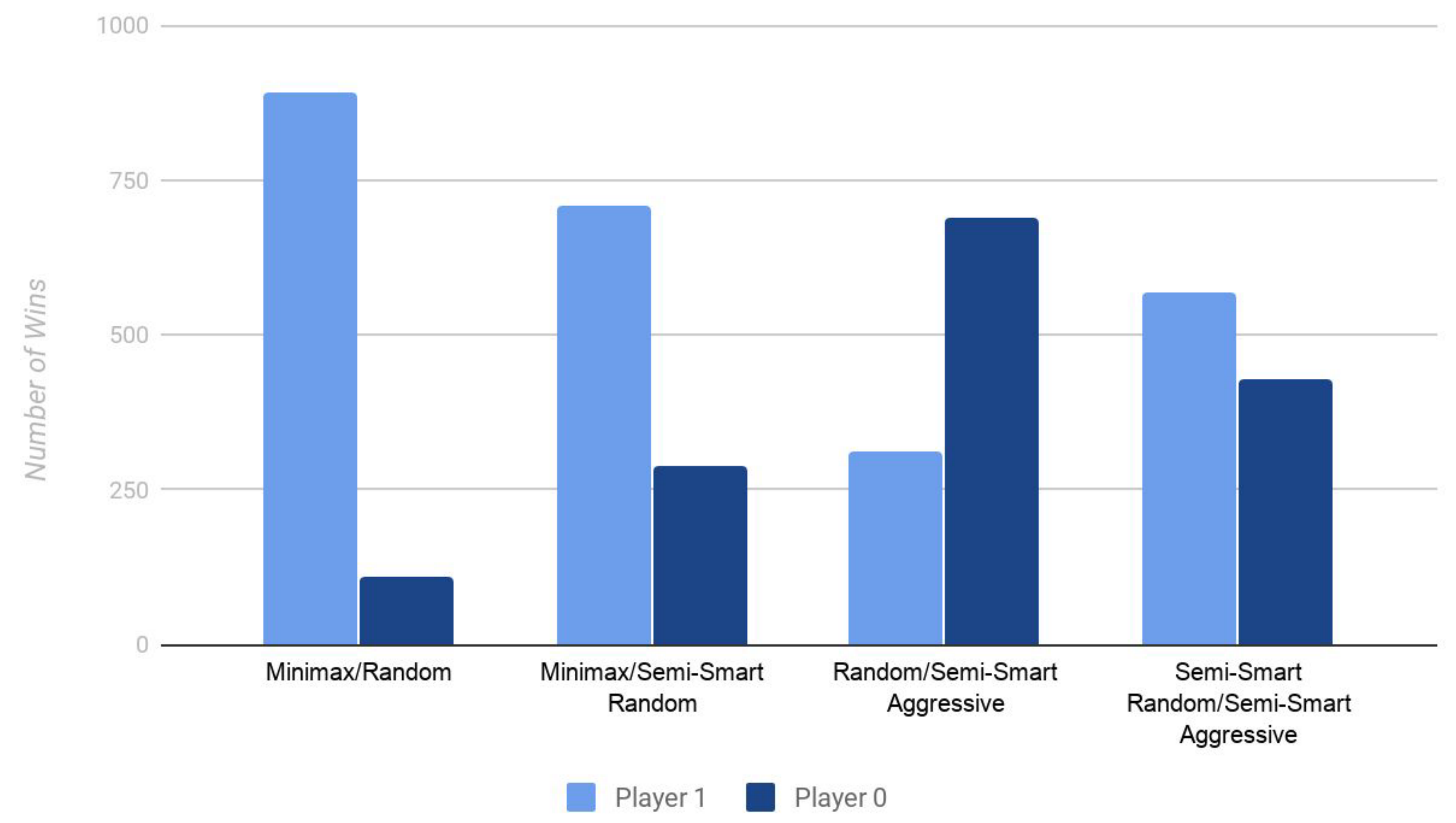


RESULTS

- Evaluation Metric: Number of games won in 1000 simulations
- We used a simple country map to be able to run the algorithm for many times
- Random initializations


Simplified Country Map

- Challenges: For Expectiminimax, large search space due to huge branching factor make search expectiminimax complexity overwhelmingly large. So it is not practical to search more than one layer which makes the algorithm myopic



Analysis

- Taking the number of dice into account seems to be the most useful information for the agents in making good decisions
- The random/semi-random agents perform better than expected against their aggressive counterpart. The minimax agent performs the best

NEXT STEPS

- Reflect all real aspects of the game
- Incorporate the fortifying phase of the player's turn
- Use other algorithms such as TD learning
- Improve data structures to make the algorithm faster
- Test the game on more complicated maps