Narahari Bharadwaj, Class of 2019

Advisor: James Glenn

CPSC 490 Project Proposal

Building a Reinforcement Learning Agent to Play the Game of Risk

Introduction and Rules of Risk

Risk is an immensely popular board game developed by Albert Lamorisse in 1957, and is a turn-based strategy game for 2-6 players that involves diplomacy with the objective of conquering all the territories on a map. The game is played on a map containing 6 continents, broken up into 42 territories. Each player has a full army, consisting of the following pieces: infantry (worth 1 army), cavalry (worth 5 armies), and artillery (worth 10 armies). The game also consists of 42 cards that are labeled with a territory and contain a picture of infantry, cavalry, or artillery, as well as 2 wild cards that have pictures of all 3 piece types but no territory.

The setup phase of the game involves each player choosing a color and counting out the number of armies each player starts with (e.g. 35 for 3 players, or 30 for 4 players). All players roll a die and whoever rolls the highest number places one army anywhere. Players take turns placing armies onto unoccupied territories until all territories are claimed. Once all territories are claimed, each player places one additional armies onto a territory they already occupy, until everyone has run out of armies. The shuffled risk cards form the draw pile.

A player’s turn consists of three phases: acquiring and placing new armies, attacking, and fortifying. At the start of every turn, each player acquires armies equal to \( \left\lfloor \frac{k}{3} \right\rfloor \), where \( k \) is the number of territories they occupy and the notation denotes the floor function. In addition, total control of a continent earns extra armies (7 for Asia, 5 for North America, 5 for Europe, 3 for Africa, 2 for South America, and 2 for Australia) at the start of each turn. Furthermore, a player collects Risk cards on any turn in which they capture a territory, and can trade in a set of 3 cards that is of the same army type, of all differing army type, or 2 of one type plus a wild card. The number of armies awarded increases with the total number of previously traded-in sets throughout the game. Furthermore, if any card turned in contains a territory the player occupies, the player is awarded exactly 2
extra armies that must be placed onto that territory. The rest of the armies may be placed freely.

The second phase of a turn is the attacking phase, which is entirely optional. A player may attack a territory that is adjacent to one of their own that contains at least 2 armies, and may continue attacking for as long as they wish. The player may switch which territories they are attacking/attacking from throughout the attacking phase. The attacker rolls 1, 2, or 3 dice and the defender rolls 1 or 2 dice. The highest die of each player is compared and if the attackers’ is higher the defender loses one army from the territory under attack. Else, the attacker loses one army from the territory they are attacking from. This process is repeated for the 2nd highest die if both players rolled more than 1 die.

The third phase of a turn is the fortification stage, where the on-turn player may move as many armies as they wish from one of their territories to an adjacent one, leaving at least one army behind.

The game ends when one player captures all 42 territories on board.

**Motivation for Reinforcement Learning Approach**

Reinforcement learning (RL) may be a suitable approach for the game of Risk because the state space is extremely rich, with over $10^{47}$ states and an extremely large average branching factor [1]. RL techniques have been around for a long time, with much of the initial work in the field employing a technique known as temporal difference (TD) learning, such as Gerald Tesauro’s TD-Gammon backgammon engine [2]. Another approach, Q-learning, works by simultaneously attempting to learn the values of all states and state/action pairs by observing rewards. However, the reason RL may prove fruitful for the game of Risk is a development by the DeepMind team, who used RL techniques to build the world’s strongest Go and Chess players, both of which relied on no domain specific knowledge or expert game database. These engines used a combination of a neural network and an MCTS component, improving both components over time through self-play [3], [4]. This is the approach that appears the most promising for building a Risk RL agent as well, although I will have to look into the options more carefully as Risk has a stochastic element that neither Go or chess contains.

To familiarize myself with approaches in RL, I intend to look through resources on Q-learning, both the Alpha Zero and AlphaGo Zero papers, and David Silver’s course at UCL on reinforcement learning [5]
Methodology for Agents

I intend to program a few different artificially intelligent agents. The first will be a simple alpha-beta search agent, to serve as a baseline for the reinforcement learning agents to be compared against. The second will be a Q-learning agent, as this is likely the simplest reinforcement learning agent to implement. The last will attempt to emulate the AlphaGo Zero engine and AlphaZero chess engines, with a combination of a neural network and MCTS that attempts to learn the game of Risk *tabula rasa* by self-play.

Disclaimer: As I familiarize myself with other approaches in RL, the list of agents might change.

Evaluation of Game Play

There is an extensive open-source Risk implementation built called Yura Domination [6]. I will most likely be implementing agents to play on top of this existing implementation, rather than implementing my own logical and graphical version of the game.

I will pit each of my three agents against each of the others in matches with several games of 2-player Risk. The length of each match will be determined once I gauge how lengthy and compute-intensive a single game is.

Extensions

If time permits, I may implement the following extensions, in order of importance and feasibility:

1. Varying compute power available for the RL agents and seeing how this affects performance
2. Extending to games of risk with more than 2 players
3. Attempting to extract higher order strategies from agent play (e.g. emphasis on controlling a specific continent, troop distribution strategy, focus on attack or defense)
4. Applying above algorithms to secret mission risk, which has element of imperfect information and its own set of rules

Deliverables
1. Source code for Risk agents
   a. Alpha-beta agent (Target: March 10th)
   b. Q-learning agent (Target: March 24th)
   c. Neural network + MCTS agent (Target: April 15th)
2. Sample games from the matches between agents perhaps with limited commentary
3. A paper detailing the approaches and performance of each of the agents

References