Modeling RISK Gameplay Through the Use of Markov Chains

Blake Hillier, Raven Johnson, Jiravit Moontep
blake.hillier@cgu.edu, raven.johnson@cgu.edu, jiravit.moontep@cgu.edu

Claremont Graduate University
Discrete Mathematical Modeling
Professor Lajpat-Rai Raheja
18 December 2020

Abstract

The game of RISK is an old, complicated game about war between 2-6 opponents, each seeking to control the entire world. Given its 42 territories spread across six continents, each with its own benefits and downfalls, this game requires complex thought to master. In this paper, we propose an AI on a simpler version of the game, using Markov chains to predict who will win a given war, and by how many units, and the Monte-Carlo Tree Search to find the best sequence of wars for both territory acquisition and maintaining a good defensive position.

1 The Game of *RISK*

RISK is a game of strategy and conquest based on moves determined by dice rolls. Players attempt to occupy each of 42 territories on a map of six continents. Each player begins equipped with an army and possessing one territory. Every turn consists of three phases: deployment, attack, and transport. During deployment, the players receive additional armies according to the number and type of territories they hold. At the attack phase, each player has the opportunity to attack an adjacent territory; if the territory is occupied, then the two armies go to war. The outcome of the war is dictated by standard six-sided dice rolls executed by both parties. Either the attacker wins and is able to occupy the territory (and subsequently move additional armies into it) or the attacker loses the war and all their armies in that territory. During transport, the players move their armies among their own territories (avoiding paths through territories controlled by other players) to fortify their defensive positions in anticipation of any attacks. Any player who loses all of their territories is eliminated. The victor is the last player standing.

Now we focus further on the war aspect of RISK. A war ensues when one player attacks another player's territory with the intent to capture said

territory. The outcome of the war is dictated by dice rolls with the attacker rolling up to three dice and the defender rolling up to two dice. The players roll the same number of dice as armies that they possess – 1 die for 1 army, 2 dice for 2 armies (or more if the player is defending), 3 dice for 3 or more armies (if the player is attacking). Because the number of dice rolled on either side may be different, we only consider the highest rolls. Thus, if the attacker is rolling 3 dice, and the defender is rolling 2, then we only consider the attacker's 2 highest rolls. We compare each party's highest rolls and then their second highest rolls. If the attacker's roll is higher, then the defender loses one army. Otherwise, the attacker loses one army. The attacker also has the option to withdraw before the war is over. After each roll the attacker decides if they want to continue the war. The attacker can continue the war provided they have more than one army in their territory and the defender has at least one army in their territory. If the defender loses all their armies, then the attacker can move anywhere from one army to one less than their total number of armies in the attacking territory. Because of this, it is important to note that the attacker can only wager one less than the total number of armies that they have in the attacking territory when selecting the number of dice that they roll. Therefore, we have three outcomes of a war: the attacker withdraws, the attacker loses the war and their attacking armies, or the attacker wins the war and the defending territory (while possibly also losing some number of armies) (Wikipedia contributors).

2 Markov Chains & War

Before we begin to explore a linear algebraic representation of a war in RISK and the set of all wars that comprises the game in its entirety, we must understand the concepts involved. A Markov chain is a stochastic model consisting of states and the probabilities of moving between these states. Markov chains have transition matrices P whose components p_{ij} are the

probabilities of transitioning from state i to state j. As such, the rows of P add to 1. These are incredibly useful for modeling deterministic games such as RISK, where each new state only depends on the previous one and is reached only after one action has occurred.

2.1 Implementation of Markov Chains

For RISK in particular, we will be representing each war as a Markov chain with states $X_n = (a_n, d_n)$ where a_n and d_n represent the number of attacking and defending armies remaining in the defending territory, respectively, after the nth die roll; we let the initial state at the beginning of the war be represented by $X_0 = (A, D)$. Any state where a_n or d_n is equal to 0 is an absorbing state, signaling the end of the war. All other states with both a_n and d_n nonzero are called transient states. There are a total of $A + D$ absorbing states and AD transient states. Our Markov chain P is a square $(AD + (A + D)) \times (AD + (A + D))$ block matrix

$$P = \begin{bmatrix} Q & R \\ 0 & I \end{bmatrix}$$

where Q gives the probabilities of transitioning from one transient state to another, R gives the probabilities of transitioning from a transient state to an absorbing state, the zero block gives the probabilities of transitioning from an absorbing state to a transient one (never possible), and the identity block gives the probabilities of transitioning from an absorbing state to itself (always 1) (Tan).

The most important questions here are: 1) What is the probability p_{ijk} of the defender losing k armies when rolling j dice against the attacker's i dice? 2) What is the probability of winning a war? 3) What is the expected number of armies the attacker and defender will lose?

2.1.1 Probabilities p_{ijk}

For the first question, we consider the fact that each positive integer in $[1, 6]$ has probability of being rolled $\frac{1}{6}$. We denote Y_1, Y_2, and Y_3 as the values of the attacker's 3 dice rolls, W_1 and W_2 as the values of the attacker's 2 dice rolls, and Z_1 and Z_2 as the values of the defender's 2 dice rolls. These random variables are assumed to be ordered such that

$$Y_1 \geq Y_2 \geq Y_3$$
$$W_1 \geq W_2$$
$$Z_1 \geq Z_2$$

For the attacker (or the defender with Z_1 and Z_2) rolling two dice, we have joint distribution of the two outcomes

$$prob(W_1 = w_1, W_2 = w_2) = \begin{cases} \frac{1}{36} & , \quad w_1 = w_2 \\ \frac{2}{36} & , \quad w_1 > w_2 \\ 0 & , \quad else \end{cases} \tag{1}$$

Then we have the probability (via marginal distribution) of the attacker's best roll W_1

$$prob(W_1 = w_1) = \begin{cases} \frac{2w_1 - 1}{36} & , \quad w_1 \, \epsilon \, [1, 6] \end{cases} \tag{2}$$

For the attacker rolling 3 dice, we have joint distribution of the best and second-best outcomes

$$prob(Y_1 = y_1, Y_2 = y_2) = \begin{cases} \frac{3y_1 - 2}{216} & , \quad y_1 = y_2 \\ \frac{6y_2 - 3}{216} & , \quad y_1 > y_2 \\ 0 & , \quad else \end{cases} \tag{3}$$

Then we have the probability (again via marginal distribution) of the attacker's best roll Y_1

$$prob(Y_1 = y_1) = \begin{cases} \frac{3(y_1)^2 - 3y_1 + 1}{216} & , \quad y_1 \, \epsilon \, [1, 6] \end{cases} \tag{4}$$

Now, the events considered here are not independent; for example, $prob(Y_1 > Z_1)$ and $prob(Y_2 > Z_2)$ are not independent since we have ordered the outcomes such that $Y_1 \geq Y_2$. Then the probability of the defender losing 2 armies when rolling 2 dice against the attacker's 3 dice is

$$\begin{aligned} p_{322} &= prob(Y_1 > Z_1, Y_2 > Z_2) \\ &= \sum_{z_1=1}^{5} \sum_{z_2=1}^{z_1} prob(Y_1 > z_1, Y_2 > z_2) prob(Z_1 = z_1, Z_2 = z_2) \qquad (5) \\ &= 0.372 \end{aligned}$$

So we have the full Markov chain P with $(AD + (A + D))^2$ components p_{ijk} of the defender losing k armies when rolling j dice against the attacker's i dice; these probabilities are found using the joint and marginal distributions given above with $i \, \epsilon \, [1, 3]$, $j \, \epsilon \, [1, 2]$, and $k \, \epsilon \, [0, D]$.

2.1.2 Winning or Losing

For the second question, we order the components in P representing the absorbing states, (A, D) with either $A = 0$ or $D = 0$, such that states $(0, 1), (0, 2), ..., (0, D)$ precede states $(1, 0), (2, 0), ..., (A, 0)$. We also assume that the transient states are ordered such that (A, D) is the last transient state of P. We denote

$$f_{ij}^{(n)} = prob(X_n = j, \; X_k \neq j, \; k \, \epsilon \, [1, n-1])$$

as the probability of reaching absorbing state j for the first time in n steps from initial transient state i at the nth step. This means we start at some transient state i and keep transitioning between transient states for $n - 1$ steps until finally reaching absorbing state j at step n. The $AD \times (A + D)$ matrix of these probabilities is $F^{(n)} = Q^{n-1}R$, since we travel among the components of Q $n - 1$ times and the components of R once. Then we have the probability of reaching absorbing state j from transient state i in any number of steps as

$$f_{ij} = \sum_{n=1}^{\infty} f_{ij}^{(n)}$$

and the $AD \times (A + D)$ matrix of these probabilities as

$$\begin{aligned} F &= \sum_{n=1}^{\infty} Q^{n-1} R \\ &= (I - Q)^{-1} R \end{aligned}$$

The initial state of a battle is always (A, D), so state i is the ADth (last) row of matrix F. The last A columns of F represent the absorbing states $(1, 0), (2, 0), ..., (A, 0)$ where the attacker wins, and the first D columns of F represent the absorbing states $(0, 1), (0, 2), ..., (0, D)$ where the defender wins. Therefore, the sum

$$\sum_{j=1}^{D} f_{AD,\ j}$$

is the probability the defender wins, and

$$\sum_{j=D+1}^{D+A} f_{AD,\ j}$$

is the probability the attacker wins.

2.1.3 Expected Losses

For the last question, we denote the losses accrued by the defender and attacker as L_D and L_A, respectively. It follows that their remaining armies are $R_D = D - L_D$ and $R_A = A - L_A$. Since either $R_D = 0$ or $R_A = 0$ at the end of a war, their probability distributions can be found from the probabilities $f_{AD,\ k}$ in the ADth row of matrix F:

$$prob(R_D = k) = \begin{cases} f_{AD,\ k} & , \quad k\ \epsilon\ [1, D] \\ 0 & , \quad else \end{cases}$$

$$prob(R_A = k) = \begin{cases} f_{AD,\ D+k} & , \quad k\ \epsilon\ [1, A] \\ 0 & , \quad else \end{cases}$$

Then the expected loss incurred by the defender is

$$\begin{aligned} \mathbf{E}(L_D) &= \mathbf{E}(D - R_D) \\ &= D - \mathbf{E}(R_D) \\ &= D - \sum_{k=1}^{D} k(f_{AD,\ k}) \end{aligned}$$

and the expected loss incurred by the attacker is

$$\begin{aligned} \mathbf{E}(L_A) &= \mathbf{E}(A - R_A) \\ &= A - \mathbf{E}(R_A) \\ &= A - \sum_{k=1}^{A} k(f_{AD,\ D+k}) \end{aligned}$$

(Osborne)

3 Monte-Carlo Tree Search

While we can use Markov Chains to model a war and predict both who will win and by how large a difference in armies, the attack phase of one player's turn can consist of multiple wars, some of which could happen in any order. A good approach would be to calculate every possible ordering of all possible wars in an attack phase, then evaluate the player's position after each step in a certain ordering, and then select the best suborder based on the player's

position. Of course, this is computationally slow, so we instead use the Monte-Carlo Tree Search (MCTS). In this section, we will simply talk about using MCTS in the context of winning a game. MCTS is highly successful when game trees become large or are computationally expensive to compute. MCTS approximates the game tree by only creating the sections it is actually searching while simulating the rest through Monte-Carlo simulations. This effectively creates a probability distribution of the possible states of the game. MCTS accomplishes this through four steps:

1. Selection
   Starting with the current state of the game, also called the HEAD, the algorithm proceeds to select children using some heuristic such as UCT or UCB1 until it reaches either a terminal state or a state with unexplored children.

2. Expansion
   Provided the current state is not terminal, the algorithm then creates a new child with uniform probability.

3. Simulation
   MCTS then simulates the rest of the game, randomly making choices for all players with uniform probability until a terminal state is reached.

4. Backpropagation
   The results of the simulated game, commonly whether the AI player has won or lost, is stored in each state along the path back to the HEAD, as is the number of simulations that state has received.

Once enough trials/time has passed, MCTS finally selects the child of the HEAD with the best results.

3.1 UCB1

We mention in the description of the selection phase that MCTS uses some heuristic to decide how it will explore the game. A commonly used heuristic is UCB1

$$UCT = \bar{X} + C\sqrt{\frac{2\ln(n)}{n_k}} \tag{6}$$

where $\bar{X}$ is the probability of success from the current state, $C > 0$ is an exploration constant, n is the number of visits to the current state, and n_k is the number of visits to the kth child. This function immediately favors states with good chances for success with $\bar{X}$, while $\sqrt{\frac{2\ln(n)}{n_k}}$ grows when n is large and n_k is small. This is regulated by the exploration constant C, resulting in a fair heuristic for deciding which states should be explored (B. et al.; Department).

4 Overview of Our RISK AI

Normally the MCTS algorithm alone is successful for deterministic games, but RISK presents an interesting problem because it has a significantly large, if not infinite, state space. To tackle this, we designed different AI's for each phase in a players turn since we believe a player is trying to maximize slightly different goals in each phase:

1. Deployment
   This is the first phase of a players turn, when they receive troops based on the cards, number of territories, and continents they control. This phase allows the player to prepare for their attack phase, and our AI accomplishes this by calculating all possible wars with their current set of territories and averaging their success rate, reinforcing the territories with the lowest chance of success. The probabilities are updated after each reinforcement to account for the various changes in the success

rates of each territory.

2. Attack
   The goal during the attack phase is two-fold: to gain as much territory as possible while still maintaining a good defensive position. In order to accomplish this, we could simply select all wars with a success rate above some threshold, but the order of the wars can change the overall defensive position of the AI. We solve this problem by using the MCTS, and we let the result be the average chance of surviving an attack as a defender. The MCTS is allowed to run for some number of seconds before following the branch with the best results.

3. Transport
   This final phase gives the player a chance to move some units from one territory to another connected territory. This can be solved by simply moving units from the territory with the highest chance of surviving an attack to the territory with the lowest chance, moving enough units to make them roughly equal. The problem here is that these territories must be connected, so we decided to find two paths, one oriented on the territory with the highest success rate, and one with the lowest territory. We then find the best transport option for each grouping and select the one which has the greatest change in success rates.

We note here the following simplifications to the rules of RISK made to make the initial AI easier to implement and understand:

- At the beginning of the game, instead of each player selecting which territories they start with, the territories are randomly assigned.
- Cards are not utilized.
- The AI does not consider the bonus armies given for controlling entire continents, though they do still gain armies from controlling entire continents.

5 Results

The AI was implemented in Python, and, in order to save time with the implementation of the game, we used code written by github user Whysmerhill, which can be found at https://github.com/Whysmerhill/Risk, which gave us a user interface and basic game structure for RISK. While this helped us by showing how the AI was behaving as well as giving us more time with the AI implementation, the code was less than optimal, which lead to several problems as we began focusing on the finer details of the AI. Our code can be found at https://github.com/BHill96/Risk. You can watch three AI's play, with the game pausing for a couple seconds after each turn to see how each AI is behaving. The AI players are able to reach the end of the game, but, since they are not trying to maximize the number of armies they can gain in the next deployment, the playthrough does take a while. There are also some implementation details which slow the game. Much of the time the game will simply restart the phase, but it can sometimes get stuck in a phase, which requires a restart. And, on occasion, the program will crash due to a bad value found in the code.

6 Conclusion

Throughout this project we learned how to utilize Markov chains to create AI opponents in the game RISK. This AI can make intelligent decisions on a simplified version of RISK and, with a little more work, could eventually play the true game with all its complexities. What remains to be answered is: How smart is our AI? Initial testing could consist of our AI competing against a simpler bot which chooses its actions following a uniform distribution. Another important data point to consider would be the optimal threshold for the attack phase. Eventually, we would want to test this AI against other AI's, but since most AI's that currently exist are proprietary or would require us to implement them, we should wait until the first two questions

are answered.

References

[B+12] Browne C. B. et al. “A survey of Monte Carlo tree search methods. IEEE Transactions on Computational Intelligence and AI in Games”. In: (2012). URL: https://doi.org/10.1109/TCIAIG.2012.2186810.

[Dep] Swarthmore CS Department. URL: https://www.cs.swarthmore.edu/~meeden/cs63/s19/reading/mcts.html.

[Osb03] Jason A. Osborne. “Markov Chains for the RISK Board Game Revisited”. In: *Mathematics Magazine* 76 (2 2003), pp. 129–135. URL: https://www4.stat.ncsu.edu/~jaosborn/research/osborne.mathmag.pdf.

[Tan97] Baris Tan. “Markov Chains and the RISK Board Game”. In: *Mathematics Magazine* 70 (5 1997), pp. 349–357. URL: https://www.researchgate.net/profile/Baris_Tan/publication/266313658_Markov_chains_and_the_RISK_board_game/links/5554b1a708ae6fd2d821b4cf/Markov-chains-and-the-RISK-board-game.pdf.

[Wik20] Wikipedia contributors. “Risk (game) — Wikipedia, The Free Encyclopedia”. In: (2020). [Online; accessed 13-November-2020]. URL: https://en.wikipedia.org/wiki/Risk_(game).