Risk Board Game Strategy Assessments

Joseph Richardson, Jacob Meline, Matthew Dayley

May 3, 2014

1 Background

"The individualist without strategy who takes opponents lightly will inevitably become a captive."[1]

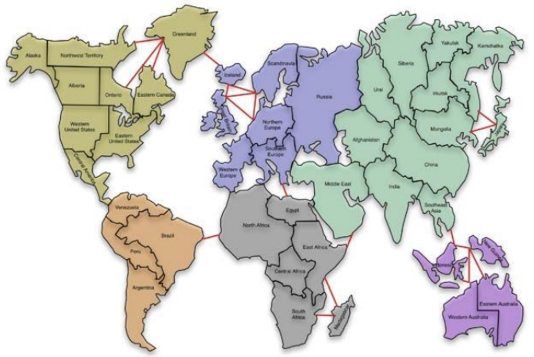
1.1 History

Risk was originally produced by parker brothers, now a division of hasbro. It was invented by the French film director Albert Lamorisse in 1957 and it was originally called "The Conquest of the World."[2]

The creator of the game anticipated that it should take 90 minutes to complete, but many players end up with the board occupying their tables for days at a time.[3] With varying strategies and fragile alliances, Risk can be a drawn-out, yet intellectually fulfilling game.

1.2 Rules

Risk is a turn-based for two to six players. In the standard Risk Game, Earth has six continents and 42 regions.



[4]

At the start of the game, players choose countries into which they place one army. This is the claiming phase. Only one army may be placed in any region during the claiming phase. At the beginning of each player's turn (after the claiming phase) they will receive bonus troops for continents that they own completely.

After claiming, players will then place armies into regions that they own. This happens, again, one at a time until everyone has placed all their starting armies.

Then players may choose to attack other regions. They may only attack bordering regions. Attackers can use up to three dice, if they have at least three troops. Otherwise, they may only use one die for each troop. Defenders roll up to two dice. The attacker and the defender then sort their dice and compare them. (Attacker's highest vs defender's highest AND attacker's second highest vs defender's second highest) For each die that is greater in this comparison, that player kills one opposing army in the battle. Defenders win ties.

In most versions of Risk, there is some sort of card system. Players could trade in cards for armies or attempt to accomplish some secret mission as dictated by the cards for an even larger army bonus. To simplify the problem that we were approaching, we decided to omit the use of cards in our project.

2 Project Scope

There are many, many strategies for risk. What we have worked to do in this project is to compare several of those strategies: Some strategies work exceptionally well in one circumstance but perform poorly in another. Which strategies work against which, and is there actually any absolute winner? How likely are the "best" algorithms to actually win? Are there loops in the "dominion graph"? Does it matter who gets the first move? These are some of the questions that we seek to answer.

In order to achieve all of this, we're going to need a LOT of data–thousands of runs for every condition-identical game we wish to assess–and each game may take a while to run. Since there are so many possible combinations of strategies, which can be permutated for many different turn-orders, and since there can be anywhere between two and six players, this project demands parallel processing.

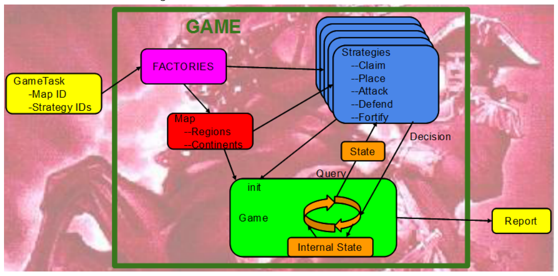
While we have said what our project goals are, it is also worth mentioning what they are NOT. We do not intend to develop any novel strategies, but rather we will stick to those we read about or have heard about (though our interpretation of known strategies may be a bit distinct). We will not be doing brute-force "look ahead". We will not be considering alternate versions of the game rules. Finnally, we are not considering the psychologocal aspects of Risk–even though, in 3+ player risk, politics is a large part of strategy. However, much of the code in our project could obviously be re-cycled for such purposes.

3 Organization and Design

3.1 Game Organization

We treat each match/game as an atomic event. The basic structure of a single game is shown here.

In order to allow remote initilization of games, we provided factory methods to generate the appropriate map and strategies based on the identifiers found in a GameTask object.

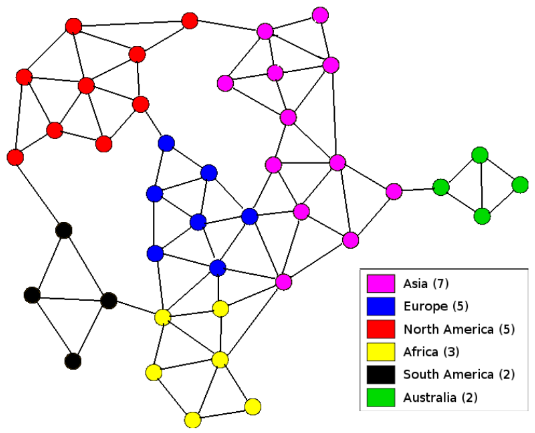


A game, once initiliazed, runs by repeatedly quiring the players' strategies to select actions and then by carrying out the requested action (for example, by using precalculated probabilities to simulate dice rolls and handle combat between regions). We took a simple object-oriented approach to designing what a strategy would be. We designed an interface that each strategy would inherit from. All strategies would have to implement four functions:

1. claim() – used at the start of the game to claim land
2. place() – used at the start of a player's turn to handle reinforcements
3. attack() – used repeatedly during a player's turn (until they player indicates that he/she is finished) to tell the Game which regions to make fight
4. fortify() – used at the end of a player's turn to move troops around

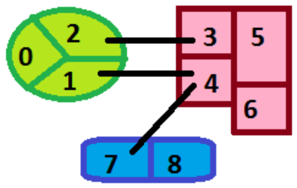
By implementing those four functions, each of which receives a copy of the game's current state, countless unique strategies may be formed. It is perfectly acceptable for each strategy to maintain an internal state, in order to track objectives, save time, or monitor trends. However, the strategies we implemented utilized only a very basic internal state. In several cases, our strategies included adjustable parameters that could be tweaked before launch (or by an inheriting class) in order to quickly form many new strategies based on one. While we provided this feature, we did not include it in our results graphs, as there were enough strategies implemented already. A discription of each strategy will be given below.

Besides a GameState (which gives the ownership and troopcount of each region), both the strategies and the Game itself require a GameMap that defines connectivity between regions, as well as continents. This was easy to represent as a list of edges forming an undirected graph, plus a list of "continents", each of which is a list of included nodes plus a value to be gained by owning all of those nodes.



[4]

In addition to the primary map of earth, we used two others in testing, one of which is the "Three-Continents" map shown here. Such maps are useful not only for testing, but for alternate analysis of strategies: Do those strategies that work well on the "Earth" map work well in general? However, while our program is fully equipped to answer this question, there was not time to re-gather all of the data for additional maps such as this.

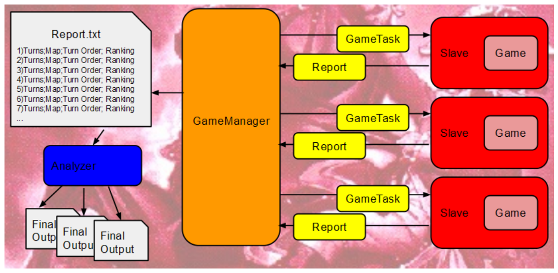


3.2 Overall Organization

We could not come to any conclusions with any desirable degree of confidence without obtaining a sizeable dataset. The large number of permutations of strategies (permutated, not merely combined, because player order can make a difference), as well as the large number of runs over which we must average to obtain reasonable results, demands that we employ parallelism.

We settled on a master-slave model, with the master (GameManager) issuing GameTasks to the GameSlaves and the slaves returning GameReports to the GameManager. In order to accomplish this, we serialized both the GameTask and the GameReport. The processor running the GameManager maintains a list of slaves (working and idle) and a list of incomplete tasks. After sending the initial GameTasks, the GameManager waits for reports and issues a new task each time one is received. Once all tasks are complete, a special GameTask consisting entirely of NOPLAYER strategies is used to signal to a slave that no additional work is required. By this means, the master and its slaves can all terminate.

The contents of each GameReport are the ID of the map used that round, the IDs of the strategies used, an ordered list indicating which player won, which came in second, etc., and finnally a count of the number of rounds the game took to complete. This arrangement allows for maximum flexibility when doing analysis; the same data gathered can easily be used for multiply studies. The GameManager does not do any direct analysis, but rather aggregates all reports into a single output file. We are then free to analyze it by any means desired. Genearlly, a spreadsheet is quick and sufficiently powerful for our needs.


Report.txt
1)Turns;Map;Turn Order; Ranking
2)Turns;Map;Turn Order; Ranking
3)Turns;Map;Turn Order; Ranking
4)Turns;Map;Turn Order; Ranking
5)Turns;Map;Turn Order; Ranking
6)Turns;Map;Turn Order; Ranking
7)Turns;Map;Turn Order; Ranking
...
Analyzer
Final Outp
Final Outpu
Final Output
GameManager
GameTask
Report
Slave
Game

After our first few strategies, we observed that each game ran considerably faster than expected, meaning that the overhead of MPI may have made our decision to task out each game an unprofitable one. It seems a more sensible division would be to let each slave run many games of a type and report back the "average" to spare the GameManager the effort. However, the meaning of "average" depends on the experiment being performed–information such as the order in which the players were illiminated may be useful, for example, but would be lost in such a scheme. We were willing to sacrifice some speed for the increased data. However, as more sophistocated strategies were written, we soon found that this debate was not actually important: Games with advanced strategies do indeed take a lot longer to run (and games with particularly foolish strategies may experience the same effect), and we soon saw the overhead and work involved in MPI paying off.

pricy

4 Type of Strategies

There are countless possible strategies. However, we found that writing each one took much longer than anticipated, being something of a project in and of itself. However, a sufficient number were made as to be interesting, and several of those are adjustable, quickly allowing us to multiply the types of strategies.

These discriptions are not complete, and the .cpp file for each should be inspected.

4.1 Bad Strategy

In order to test our code and make sure everything was working, of course we had to make a very simple strategy. This strategy takes the first available location it finds while claiming, always attacks a set number of times per turn (or less), and generally does other mindless things. This strategy will likely lose to every other strategy except the Pacifist. In order to keep runtimes reasonable (since BadStrategy makes no concious attempt to win or end the game), BadStrategy was not included in the final all-vs-all, but initial testing indices that does indeed perform horendously.

4.2 Immediate Best Value

This is a greedy algorithm that assigns a score to its property and always takes the move that immediately maximizes that score. When assigning value, it uses this equation:

$$V = x_1 T/B + x_2 N$$

T represents the troop bonus (due to number of regions owned and due to owned continents). B is the number of exposed borders, something that is usually sought to be minimized. However, there is also a merit in the opportunity to expant, which is represented by N, the number of neighboring regions. X1 and X2 are scaling factors that reflect the strategies desire for stability vs opporunity; these two values are left adjustable.

When attacking, the strategies does not consider only the value, but also the probability of victory; after multiplying V by this probability it may be found that it is better to attack a slightly less valuable location at which victory is more assured. Probabilities of victory are obtained via a static method of the Game class, which uses a cache to determine exact probabilities at low army counts, and approximations at higher army counts. It also considers an adjustable “dangerThreshold” to avoid leaving its borders unprotected. Reinforcements are placed near valuable-to-conquer locations in order to prepare for battle.

4.3 Smallest Continent first

A popular strategy where the player looks to claim either Australia or South America since they are easier to defend due to the limited number of paths into and out of them and due to the quick troop boost they can provide; perhaps more than any other this is a strategy whose success we are interested in guaging. It will attack rival on the smallest continent that it can reach. When placing reinforcements, is devides them (unequally, as determined by a formula involving the adjustable “attackPlacementPreferenceFactor”) between regions bordering the continent it wishes to conquer and the exposed borders of any continent already aquired, to keep its hold.

4.4 Prey on the weak

Prey on the weak does what you’d expect: It targets the weakest bordering country to get a quick kill (or a seriese of them). It will attack as long as its foes are sufficiently weak and it is attacking with no fewer than three armies. Reinforcements are placed by the weakest opponents, in an attempt to “sweet” through as much enemy territory as possible.

4.5 Pacifist

This strategy will never win because it never attacks. But it may perhaps outlive other strategies. It will be easy to determine if a strategy is far too aggressive by comparing it against this strategy. For example, if strategy A is knocked out of the game frequently before the pacifist strategy, then strategy A is likely too aggressive to be an effective strategy. When placing, claiming, and reinforcing, the pacifist strategy picks arbitrarily.

4.6 Aggressive

We have a character profile on this type: This player lost his job, failed his classes, and lost his girlfriend to his roommate. He is out for vengenance and is full out rage. Lacking any moral compass, he selects anyone to attack and will fight to the death, never turning down a fight, no matter how foolish. While this is a genearlly aimless strategy, it can overwhelm many strategies, and can be expected to shorten games in which it is involved.

4.7 Human Controlled

For debugging purposes and getting a real feel for how certain strategies will play out. This strategy uses a very simple textual user interface to display state and request user input. This strategy cannot, of course, be used in our final all-vs-all, but it provides a strong subjective measure of other strategies' values. For example, Matt played a quick game on a condensed map vs the “Smallest Continents First” strategy and lost. This provides evidence that the smallest continent first strategy can actually work in a “real” game against people. The Human Controlled Strategy is also fun.

5 Data Aquisition

5.1 Our chosen experiments

We decided on three primary experiments for this project.

First, we wished to run a simple two-player all-vs-all set of matches in order to form a “dominion graph”. By analyzing the dominion graph, we could determine whether there were any obvious winner or whether there were cycles. While a traditional dominion graph is binary and simply has an arrow on each edge, that does not accurately describe the situation here, so we include a probability alongside the arrow to indicate with what probabity the “better” of two strategies would win, since we cannot make binary sta

Second, we wished to determine if player order has an effect on triumph. For each strategy, we pitted between 2 and 6 copies against itself and saw how often the first player won vs the second, third, etc. We also use the all-vs-all data to see whether the results between two distinct strategies were affected by which played first.

Finnally, we wished to analize the number of rounds required for victory. As with experiment #2, we conisder the “vs itself” data for 2-6 players and the “vs others” data for two players. We are interested to know if all strategies prefer preemption or not, and whether a strategy that runs fast or slow vs itself does the same against others.

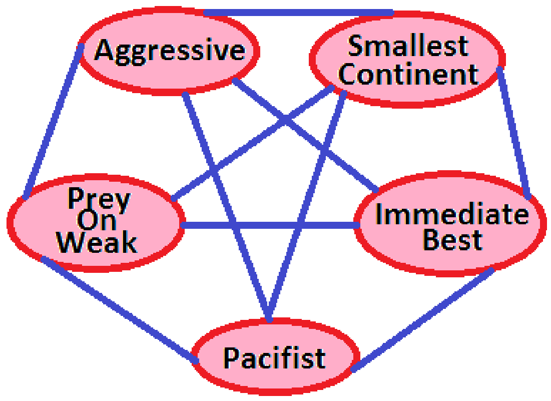
5.2 Gathering the data

Initially, we relied mostly on a test main() program to run games individually. As we became ready to use the GameManager, we needed a way to place orders for lists of tasks. To accomplish this, we wrote two generators that would take lists of strategies and maps and permutate them (with or without repetition) according to a number of players. We also used a bash script to split output between several files for merging later, in order to safely obtain large datasets without the need to precisely estimate runtime. Yet another bash script was responsible for launching the “vs self” games with every desired number of players.

Data, once ready, was fed into pre-made Excel sheets for quick analysis.

6 Results

6.1 Experiment 1: Dominion Graph



6.2 Experiment 2: Order Advantage

For details, see “Order Advantage study.xlsx”. There were several interesting trends:

1. For Smallest Continent First vs itself, being the first player is actually the very worst outcome, unless there are only two players (in which case the first player wins 66% of the time. Beyond that, it was hard to recognize a pattern.

2. For Agressive vs itself, for every number of players, the later you got your first turn the more likely you were to win, monatomically. This is very interesting, and is likely due to the fact that the agressive style spreads itself thin and as such is most vulnerable to attack right after its turn.

3. For Prey on the Weak vs itself, the first player had the advantage every time, but the advantage very quickly became negligible as the number of players increased. This is likely due to the fact that a computer can easlity get trapped in a bad pattern: Those players who did not go first will tend not to bother the first player, who got the first reinforcements, and so they will distribute their forces around other players, leaving themselves open for attack. Indeed, it seems to make no difference whether one is second player or fifth; only the first player has an advantage. Prey on the Weak is intimidated by any show of muscle. Immediate Best exhibits the very same pattern; apperantly the multiplying-value-by-probability-of-victory makes this stratigy similarly cowardly.

6.3 Experiment 3: Number of Rounds

For details, see “Order Advantage study.xlsx”. There were several interesting trends:

1. As should surprise no one, Aggressive finished the most quickly across the board and with a standard deviation of only ~35%. SmallestContinent.

2. Prey on the Weak and ImmediateBest took a LOT longer than other strategies to finish, and had a higher standard deviation percent. But in the latter’s case, the percent deviation decreases with the nuber of players. In each case, 3-player games take a lot longer than 2-player ones, and 4 takes longer than 3, but after that it’s relatively stable.

In summary, the number of players playing didn’t make as much of a difference as expected, but which strategy is used sure makes a big difference.

7 Conclusion

Writing strategies, it seems, is much harder than anticipated. However, we found our program to be very useful and interesting. Thanks to it, we are able a qualitatively measure the fitness of strategies in comparison to one another, which reveals interesting trends that one would normally never have enough data to detect.

References

[1] Sun Tzu, *The Art of War*

[2] Wikipedia contributors, "Risk (game)," Wikipedia, The Free Encyclopedia, accessed April 26, 2014, http://en.wikipedia.org/wiki/Risk_(game).

[3] Keith Veronese, "The Origins and Evolution of the Strategy Board Game RISK," last modified March 30, 2012, http://io9.com/5897532/the-origins-and-evolution-of-the-strategy-board-game-risk.

[4] Garrett Robinson, "The Strategy of Risk," accessed April 26, 2014, http://web.mit.edu/sp.268/www/risk.pdf.