



GRADUATE SCHOOL
EAST TENNESSEE STATE UNIVERSITY

East Tennessee State University
Digital Commons @ East
Tennessee State University

Electronic Theses and Dissertations


Student Works

5-2022

RISK Gameplay Analysis Using Stochastic Beam Search

Jacob Gillenwater
East Tennessee State University

Follow this and additional works at: <https://dc.etsu.edu/etd>

 Part of the [Artificial Intelligence and Robotics Commons](#), [Software Engineering Commons](#), and the [Theory and Algorithms Commons](#)

Recommended Citation

Gillenwater, Jacob, "RISK Gameplay Analysis Using Stochastic Beam Search" (2022). *Electronic Theses and Dissertations*. Paper 4063. <https://dc.etsu.edu/etd/4063>

This Thesis - embargo is brought to you for free and open access by the Student Works at Digital Commons @ East Tennessee State University. It has been accepted for inclusion in Electronic Theses and Dissertations by an authorized administrator of Digital Commons @ East Tennessee State University. For more information, please contact digilib@etsu.edu.

RISK Gameplay Analysis Using Stochastic Beam Search

A thesis
presented to
the faculty of the Department of Computing
East Tennessee State University

In partial fulfillment
of the requirements for the degree
Master of Science in Computer Science, concentration in Applied Computer Science

by
Jacob Thomas Gillenwater
May 2022

Dr. Brian Bennett, Chair
Dr. Ghaith Husari
Dr. Jeff Roach

Keywords: Stochastic Beam Search, Genetic Algorithm, Artificial Intelligence, RISK, Strategy Games

ABSTRACT

RISK Gameplay Analysis Using Stochastic Beam Search

by

Jacob Thomas Gillenwater

Hasbro's RISK, first published in 1959, is a complex multiplayer strategy game that has received little attention from the scientific community. Training artificial intelligence (AI) agents using stochastic beam search gives insight into effective strategy when playing RISK. A comprehensive analysis of the systems of play challenges preconceptions about good strategy in some areas of the game while reinforcing those preconceptions in others. This study applies stochastic beam search to discover optimal strategies in RISK. Results of the search show both support for and challenges to traditionally held positions about RISK gameplay. While stochastic beam search competently investigates gameplay on a turn-by-turn basis, the search cannot create contingencies that allow for effective strategy across multiple turns. Future work would investigate additional algorithms that eliminate this limitation to provide further insights into optimal gameplay strategies.

Copyright 2022 by Jacob Thomas Gillenwater
All Rights Reserved

DEDICATION

I dedicate this thesis to my family, both immediate and extended. I am certain that without the loving support, encouragement, and understanding, this thesis would not have been completed.

ACKNOWLEDGEMENTS

I would like to thank Dr. Brian Bennett for his continual guidance and support during my tenure at East Tennessee State University. It was through his classes that I arrived at this research topic. As an advisor, he pushed me to my limits and beyond to accomplish more than I thought myself capable.

I would like to acknowledge the collective encouragement, wisdom, and assistance from my many professors at East Tennessee State University and Northeast State Community College. I would also like to acknowledge many of my fellow graduate students for their support, encouragement, and solidarity during many late nights.

Finally, I must thank my supportive family. They have stood by me since birth and have watched me through many long days, tough classes, and all-night study sessions. I would not be a person capable of completing such research without their help. Unfortunately, six of those who saw the start of my collegiate journey did not live to see the end: Beulah “Boots” Sykes, Albert Lay Jr., Mary Calhoun, Ronald Sykes, Edmond “Wayne” Jennings, and Coy Gillenwater. I will miss their love, guidance, and wisdom for the rest of my life. Lastly, with all of my heart, I am thankful for the love and support of Sabrina Gillenwater, Jamey Gillenwater, Rusty Gillenwater, Xander Hooker, Gary Calhoun, Sue Gillenwater, and many other aunts, uncles, and cousins. None of my achievements would have been possible without their help, and I am forever grateful for each of them.

TABLE OF CONTENTS

ABSTRACT.....	2
DEDICATION.....	4
ACKNOWLEDGEMENTS.....	5
LIST OF TABLES.....	10
LIST OF FIGURES.....	11
LIST OF EQUATIONS.....	15
Chapter 1 Introduction.....	16
1.1 Overview and Problem Statement.....	16
1.2 Purpose of the Study.....	17
1.3 How to play RISK.....	17
1.3.1 Game Setup.....	17
1.3.2 Placement Phase.....	22
1.3.3 Attack Phase.....	23
1.3.4 Movement Phase.....	25
1.3.5 RISK Cards.....	25
1.3.6 Game Ending Conditions.....	26
1.3.7 Rule Modifications.....	26
1.3.7.1 Tie-Breaking Rules.....	27
1.4 Stochastic Beam Search.....	28
Chapter 2 Literature Review.....	30
2.1 Blogger Strategies.....	30
2.2 Attack Strategies.....	31
2.3 Wolf’s Implementation of Artificial Intelligent RISK Agent.....	33
2.3.1 Basic Behavior Agent.....	33
2.3.2 Advanced Behavior Agent.....	34
2.3.3 Conclusions.....	35
2.4 Erik Blomqvist’s Implementation of an AlphaZero Agent.....	36
2.4.1 Conclusions.....	37
2.5 Stochastic Beam Search.....	37
Chapter 3 Approach.....	39

3.1	High-Level Overview	39
3.2	Agents	39
3.2.1	Agent Characteristics	40
3.2.2	Agent Mutation	41
3.3	RISK Agents	41
3.3.1	Placement Phase.....	42
3.3.2	Attack Phase.....	45
3.3.2.1	Attack Preparation	45
3.3.2.2	Attack Estimates	46
3.3.2.3	Attack Impact	47
3.3.3	Movement Phase.....	49
3.3.3.1	Quantifying Risk of Movement.....	49
3.3.3.2	Movement Impact.....	51
3.4	Attack System	53
3.4.1	Simulating Attacks.....	53
3.5	Using Estimates to Make Predictions	55
3.5.1.1	Simple Estimates	55
3.5.1.2	Elaborate Estimates	55
3.6	Map	56
3.6.1	Territories.....	57
3.6.2	Continents	58
3.6.3	Maps as a Game State	59
3.6.4	Serializing Map Data	59
3.7	A Game of RISK.....	60
3.7.1	Game Setup.....	61
3.7.2	Game Play.....	61
3.7.3	Game Over	62
3.7.3.1	Tiebreakers	62
3.8	Agent Population	63
3.8.1	Initiating a Population.....	63
3.8.2	Creating a New Generation of Agents	64
3.8.3	Average Agent	64

3.8.4	Dividing Agents into Matchups	65
3.9	Tournament Play	65
3.9.1	Organizing a Tournament	66
3.9.2	Tournament Series	67
3.10	Runtime Optimizations	68
3.10.1	Parallel Processes	68
3.10.2	Data Caching	68
3.10.3	Limited Dice Rolls	69
3.11	Error Recovery	69
Chapter 4	Results	71
4.1	Placement Category	71
4.2	Place Continent Category	72
4.3	Attack Category	73
4.4	Dice Roll Analysis	74
4.5	Attack Continent Category	75
4.6	Movement Category	76
4.7	Preference Category	77
Chapter 5	Conclusions & Future Research	78
5.1	Comparison to Blogger Strategies	78
5.2	Future Research	79
REFERENCES	80
APPENDICES	81
Appendix A:	Agent Definition	81
Placement Category	81
Place Continent Category	82
Attack Category	83
Attack Continent Category	84
Movement Category	85
Preference Category	86
Appendix B:	Results Graphs	87
Placement Category	87
Place Continent Category	93

Attack Category	99
Attack Continent Category	111
Movement Category.....	117
Preference Category	127
Appendix C: Dice Analysis Results.....	131
VITA.....	137

LIST OF TABLES

Table 1. Initial army counts and the number of possible starting positions for each allowable number of players on a standard RISK game board.	20
Table 2. List of bonus armies a player receives at the start of their next turn for claiming all territories in a continent.....	23
Table 3. Key explaining how to read an Agent Characteristic chart.	81
Table 4. Agent Characteristic chart for the Placement category.	81
Table 5. Agent Characteristic chart for the Place Continent category.....	82
Table 6. Agent Characteristic chart for the Attack category.	83
Table 7. Agent Characteristic chart for the Attack Continent category.....	84
Table 8. Agent Characteristic chart for the Movement category.....	85
Table 9. Agent Characteristic chart for the Preference category.	86

LIST OF FIGURES

Figure 1. A typical RISK game board map with each continent a different color.	18
Figure 2. A simplified RISK map represented as a network of nodes with each set in a continent a different color.	19
Figure 3. Graphical interpretation of Agent data structure.	40
Figure 4. Examples showing how movements may be classified as either "Risky" or "Safe" behavior.	50
Figure 5. Graphical interpretation of Map data structure.	57
Figure 6. Graph of how the "Ally Adjacent" characteristic from the "Placement" category changed over time.	87
Figure 7. Graph of how the "Anywhere" characteristic from the "Placement" category changed over time.	88
Figure 8. Graph of how the "Border Adjacent" characteristic from the "Placement" category changed over time.	89
Figure 9. Graph of how the "Connection Bias" characteristic from the "Placement" category changed over time.	90
Figure 10. Graph of how the "Enemy Adjacent" characteristic from the "Placement" category changed over time.	91
Figure 11. Graph of how the "Placement Bias Multiplier" characteristic from the "Placement" category changed over time.	92
Figure 12. Graph of how the "Africa" characteristic from the "Place Continent" category changed over time.	93
Figure 13. Graph of how the "Asia" characteristic from the "Place Continent" category changed over time.	94
Figure 14. Graph of how the "Australia" characteristic from the "Place Continent" category changed over time.	95
Figure 15. Graph of how the "Europe" characteristic from the "Place Continent" category changed over time.	96
Figure 16. Graph of how the "North America" characteristic from the "Place Continent" category changed over time.	97

Figure 17. Graph of how the "South America" characteristic from the "Place Continent" category changed over time.....	98
Figure 18. Graph of how the "Ally Adjacent" characteristic from the "Attack" category changed over time.	99
Figure 19. Graph of how the "Anywhere" characteristic from the "Attack" category changed over time.	100
Figure 20. Graph of how the "Attack Dice Count" characteristic from the "Attack" category changed over time.....	101
Figure 21. Graph of how the "Border Adjacent" characteristic from the "Attack" category changed over time.	102
Figure 22. Graph of how the "Capture Continent" characteristic from the "Attack" category changed over time.....	103
Figure 23. Graph of how the "Defend Dice Count" characteristic from the "Attack" category changed over time.....	104
Figure 24. Graph of how the "Minimal Remaining Percent" characteristic from the "Attack" category changed over time.....	105
Figure 25. Graph of how the "Minimal Success Chance" characteristic from the "Attack" category changed over time.....	106
Figure 26. Graph of how the "Minimum Impact Score" characteristic from the "Attack" category changed over time.....	107
Figure 27. Graph of how the "Safe Threshold" characteristic from the "Attack" category changed over time.	108
Figure 28. Graph of how the "Survivorship Bias" characteristic from the "Attack" category changed over time.....	109
Figure 29. Graph of how the "Targets Destroyed Bias" characteristic from the "Attack" category changed over time.....	110
Figure 30. Graph of how the "Africa" characteristic from the "Attack Continent" category changed over time.	111
Figure 31. Graph of how the "Asia" characteristic from the "Attack Continent" category changed over time.	112

Figure 32. Graph of how the "Australia" characteristic from the "Attack Continent" category changed over time.....	113
Figure 33. Graph of how the "Europe" characteristic from the "Attack Continent" category changed over time.....	114
Figure 34. Graph of how the "North America" characteristic from the "Attack Continent" category changed over time.....	115
Figure 35. Graph of how the "South America" characteristic from the "Attack Continent" category changed over time.....	116
Figure 36. Graph of how the "Ally Adjacent" characteristic from the "Movement" category changed over time.....	117
Figure 37. Graph of how the "Anywhere" characteristic from the "Movement" category changed over time.	118
Figure 38. Graph of how the "Bigger Territory" characteristic from the "Movement" category changed over time.....	119
Figure 39. Graph of how the "Border Adjacent" characteristic from the "Movement" category changed over time.....	120
Figure 40. Graph of how the "Connection Bias" characteristic from the "Movement" category changed over time.....	121
Figure 41. Graph of how the "Enemy Adjacent" characteristic from the "Movement" category changed over time.....	122
Figure 42. Graph of how the "Minimum Impact Score" characteristic from the "Movement" category changed over time.	123
Figure 43. Graph of how the "Risky Transfer Rate" characteristic from the "Movement" category changed over time.....	124
Figure 44. Graph of how the "Safe Transfer Rate" characteristic from the "Movement" category changed over time.....	125
Figure 45. Graph of how the "Small Territory" characteristic from the "Movement" category changed over time.....	126
Figure 46. Graph of how the "Larger" characteristic from the "Preference" category changed over time.	127

Figure 47. Graph of how the "Risky" characteristic from the "Preference" category changed over time.	128
Figure 48. Graph of how the "Safe" characteristic from the "Preference" category changed over time.	129
Figure 49. Graph of how the "Smaller" characteristic from the "Preference" category changed over time.	130
Figure 50. Heatmap of attack success chance using a maximum attackers strategy. The defender is using a maximum defenders strategy.	131
Figure 51. Heatmap of attack success chance using a minimum attackers strategy. The defender is using a maximum defenders strategy.	132
Figure 52. Heatmap of the difference between the two attack strategies. Red indicates no change.....	133
Figure 53. Heatmap of attack success chance using a maximum defenders strategy. The attacker is using a maximum attackers strategy.	134
Figure 54. Heatmap of attack success chance using a minimum defenders strategy. The attacker is using a maximum attackers strategy.	135
Figure 55. Heatmap of the difference between the two defensive strategies. Green indicates no change.	136

LIST OF EQUATIONS

Equation 1. Initial game state equation, assuming a standard map size, and that territories are evenly distributed to all players.	21
Equation 2. Initial game state equation in a generalized form.....	21
Equation 3. Simplified base army count equation	22
Equation 4. Complex base army count equation.	22
Equation 5. Equation for determining bonus army payouts when discarding RISK card sets.....	26
Equation 6. Equation for determining how many armies to move between two territories given a transfer percent.	51
Equation 7. Summation rule showing how a constant multiple of a function can be placed either inside or outside of the summation.	65

Chapter 1. Introduction

1.1 Overview and Problem Statement

When discussing artificial intelligence, the broad topic of games comes up often. But what is a game? Russell & Norvig discussed games as multiagent environments where each agent interacts with one another directly or indirectly to achieve some goal or maximize some metric [1, pp. 161-163]. Some games involve random chance elements to minimize the importance of skill and experience between agents while also introducing an element of surprise that can be enjoyable for human agents. Games that minimize elements of chance and focus on player interactions across multiple turns between agents are often categorized as strategy games.

A popular example of a strategy game is chess. In chess, two-player agents move pieces along a game board with the ultimate goal of removing a “King” piece out of play. The chess environment is fully observable, meaning there is no hidden information, deterministic, meaning there are no random elements, and zero-sum, meaning that each move that benefits one agent also harms the other [1, pp. 161-163].

The board game RISK is a very different type of strategy game compared to chess. RISK describes itself as a turn-based grand strategy game where multiple players compete for territories on the game board [2]. Albert Lamorisse designed RISK in 1957 [3]. RISK can be played with two to six players who place and control units, called armies, on a game board [2]. Each turn in RISK comes in three distinct phases: an army placement phase, an attack phase, and a movement phase.

RISK is fully observable and zero-sum like chess, but it is not deterministic because random dice rolls influence attack success. While tactical players can increase their odds of a

successful attack, the outcome is ultimately unpredictable. The unpredictability combined with billions of possible game states makes RISK a dauntingly complex game to study.

1.2 Purpose of the Study

The complexity of the problem has not stopped analysis of the game to search for new game-winning strategies. The game is far too expansive to explore by a human. This thesis aimed to analyze generations of artificial intelligence agents to support or refute commonly-held decision-making strategies and create a basis for an eventual AI agent that could play RISK at a competitive level.

After creating an environment with the rules of RISK programmed into it, I create a large population of artificially intelligent agents that compete against one another in a double-elimination tournament. Initially, agents act randomly, but from the chaos of random actions, successful agents emerge on which future generations of agents will base their choices. This process serves as the basis of the learning algorithm used by my implementation—stochastic beam search [1].

1.3 How to play RISK

1.3.1 Game Setup

Albert Lamorisse originally designed RISK as a tabletop board game, so the game state is best summarized as the state of the game board [3]. The game board is a simplified political map of the world, with each region divided up into distinct continents (Figure 1). The map may be reduced trivially to a network of nodes grouped into distinct sets, which can make the map easier to understand (Figure 2). The players can use armies to interact with the territories on the game

board's map. Additionally, players may claim ownership over any territory that holds that player's armies.

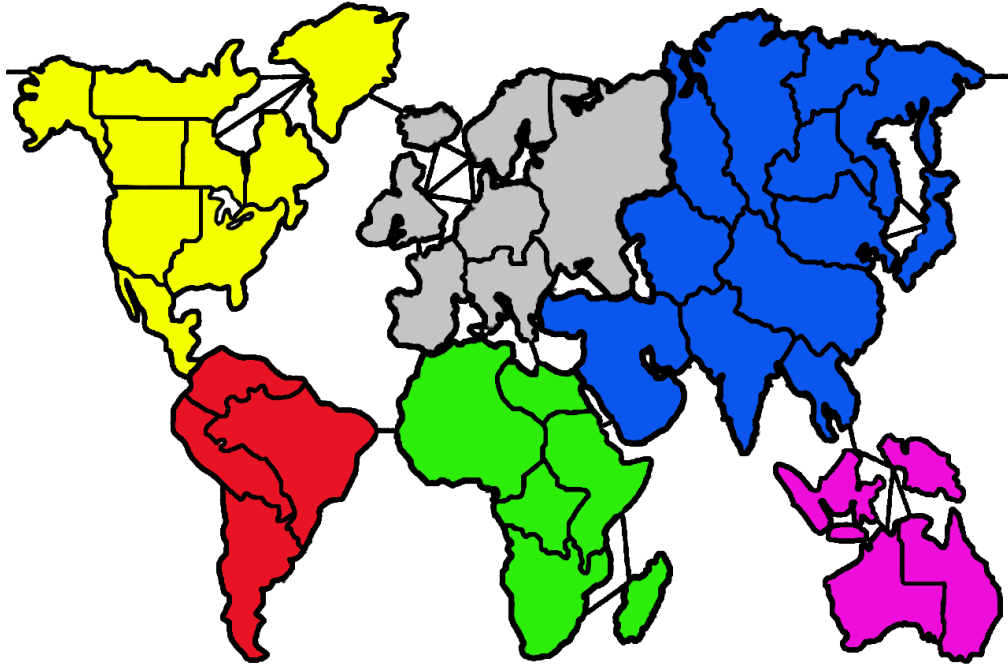


Figure 1. A typical RISK game board map with each continent a different color.

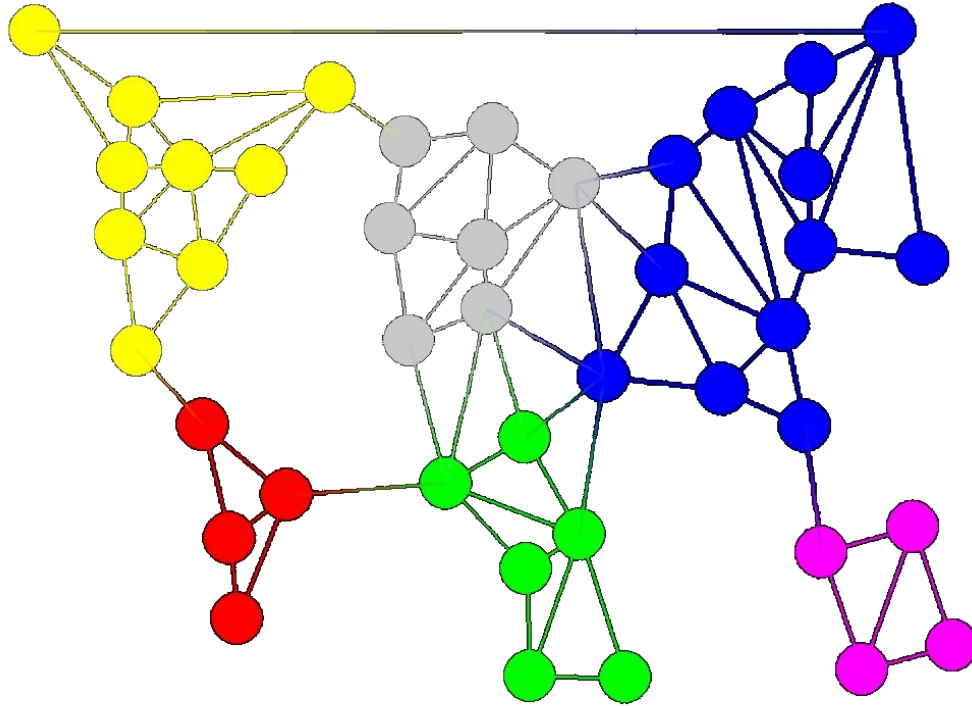


Figure 2. A simplified RISK map represented as a network of nodes with each set in a continent a different color.

RISK has an interesting pregame whereby player agents place armies around the game board in a round-robin fashion to claim territories on the board's map [2]. The players are given a predetermined number of armies to place depending on the total number of players at the start of the game (Table 1). The standard number of territories on a game board is 42.

Initial Army Counts		
Number of Players	Number of Armies for Initial Placement	Number of Possible Starting Positions
2	40	8.418×10^{156}
3	35	3.098×10^{171}
4	30	2.922×10^{165}
5	25	2.270×10^{157}
6	20	3.626×10^{152}

Table 1. Initial army counts and the number of possible starting positions for each allowable number of players on a standard RISK game board.

Turn order is randomly decided. After agreeing on the turn order, each player must place exactly one army onto an unclaimed territory to claim it before ending their turn. The players will continue in this fashion until all territories are claimed. A player may place any remaining armies onto territories already claimed by that player. Once all players have placed all of their initial armies onto the map, normal play may begin using the same turn order as during the pregame.

The range of possible initial game states is immense. First, consider the process of claiming territories. Consider a standard map (Figure 1); the first player will have 42 territories available for placement. The second player will have 41, the third will have 40, and so on, leaving $42!$ total combinations.

Next, consider that each player must now distribute their remaining armies onto their claimed territories. Consider a standard map and four players; the first and second players will have 11 options for placing their first army, eleven options for placing their second army, 11 options for placing their third army, and so on. The third and fourth players will have ten options for placing their first army, ten options for placing their second army, and so on. For simplicity, the following calculation will make the false assumption that all players receive an equal number

of territories to start. Letting a represent the total number of armies given to the player at the start of the setup phase and n represent the total number of players, we can conclude the following calculation:

$$\begin{aligned} & \text{Initial Game States Assuming} \\ & \text{Territories are Evenly Distributed} = 42! \times \left(\left(\frac{42}{n} \right)^{a - \frac{42}{n}} \right)^n \end{aligned}$$

Equation 1. Initial game state equation, assuming a standard map size, and that territories are evenly distributed to all players.

The formulation may appear complicated at first, but consider that $\frac{42}{n}$ is simply the number of claimed territories of each player. Therefore, $a - \frac{42}{n}$ is the number of armies remaining after all territories have been claimed. Unfortunately, this is not the end of the calculation because 42 is not evenly divisible by all possible values of n , i.e., 2 through 6. To account for the players that will receive either one more or one less than all other players, we can use the floor of $\frac{42}{n}$ in conjunction with modulus operations to account for the discrepancy. $42 \bmod n$ denotes the number of players that receive one additional territory because of the unfair division. Conversely, $n - (42 \bmod n)$ is the number of players that receive no additional territory. Additionally, the variable t will represent the total number of available territories on the game board to provide for the most general case. This is the equation used to complete Table 1.

$$\text{Initial Game States} = t! \times \left(\left(\left\lfloor \frac{t}{n} \right\rfloor + 1 \right)^{a - \left\lfloor \frac{t}{n} \right\rfloor} \right)^{(t \bmod n)} \times \left(\left(\left\lfloor \frac{t}{n} \right\rfloor \right)^{a - \left\lfloor \frac{t}{n} \right\rfloor} \right)^{n - (t \bmod n)}$$

Equation 2. Initial game state equation in a generalized form.

1.3.2 Placement Phase

Each turn of normal play begins with the placement phase. The engaging player is given an algorithmically generated number of armies to place on claimed territories. A player claims a territory so long as they have at least one army on the territory. To determine the base number of armies that a player can place, one simply counts the total number of territories that the player has claimed, divides that sum by three, and then rounds down to the nearest whole number if necessary.

$$Armies = \left\lfloor \frac{\sum Territories}{3} \right\rfloor$$

Equation 3. Simplified base army count equation

The above formulation computes the base number, but some additional rules will update and change that number. For example, the rules guarantee that each player must receive at least three armies. Additionally, players may receive bonus armies for controlling a specific subset of territories called continents [2]. Lastly, players may receive bonus armies by trading in RISK cards. The number of bonus armies received for controlling each continent can be found in Table 2. Bonus armies from claiming all territories within a continent are additive. This means that the final formulation for how many armies a player should receive at the start of the placement phase of their turn is as follows:

$$Armies = \max \left(\left\lfloor \frac{\sum Territories}{3} \right\rfloor + (Bonus Armies) \mid 3 \right)$$

Equation 4. Complex base army count equation.

Continent Control Bonuses	
Continent Name	Bonus Armies Received
Asia	7
North America	5
Europe	5
Africa	3
South America	2
Australia	2

Table 2. List of bonus armies a player receives at the start of their next turn for claiming all territories in a continent.

1.3.3 Attack Phase

Each turn of normal play includes an attack phase immediately following the conclusion of the placement phase [2]. During the attack phase, players may choose to attack zero, one, or many territories controlled by the other players. A player may only attack a territory directly connected to a territory they already claim. Additionally, a player may only attack with the armies available on the connected territory and must attack one territory at a time. A final constraint on a player's attack capabilities is ensuring that the attacking territory has at least one army remaining on the territory following the attack. Consequently, a territory must have at least two armies to attack a connected territory.

A player may attack many different territories during their attack phase. Additionally, a player may have many different territories attack a single territory during their attack phase. Furthermore, a player may have a territory attack the same target territory multiple times within a single attack phase. The player with armies on the defending territory must defend with at least one army whenever attacked.

Each attack between territories may be more easily conceptualized as a series of battles between an individual army or small groups of armies from the attacking and defending territories. Armies from the attacking territory are separated into groups of up to three, and armies from the defending or targeted territory are separated into groups of up to two. Each battle's outcome is decided by pairing up groups from attacking and defending territories.

Each army in a group of armies, either attacking or defending, represents a single six-sided dice roll [2]. After rolling a die for each army, the results of each group may be compared. First, the highest value of both the attacking and defending group are paired together, followed by the second highest values of each group if applicable. The value from any remaining dice in the attacking group or the defending group is discarded. When comparing pairs of values, the army with the corresponding lower value is removed from play. In the event of a tie between pairs of values, the attacking army is removed from play. Consequently, the highest amount of armies that an attacker or a defender can lose in a single attack is two.

An attack ends once the defending territory has no more armies for defense. When this occurs, all remaining armies from the attacking group of armies move onto the target territory, ensuring that at least one army remains on the attacking territory. Once the attacking player has moved their armies onto the territory, that player now controls the territory and may attack from it, move armies to it in the movement phase, or place new armies upon it in their next placement phase. The defending player loses control of the territory, and may no longer claim ownership of the territory, place new armies onto the territory, or move armies onto the territory.

1.3.4 Movement Phase

Each turn of normal play includes a movement phase immediately following the conclusion of the attack phase [2]. The movement phase is the last phase of a player's turn in normal play. During the movement phase, a player may choose to move armies from exactly one territory to another connected territory that has at least one of their armies already on it. Similar to the rule regarding attacks, which requires at least one army to remain on the attacking territory, a rule for movement requires at least one army to remain on the territory supplying the moving armies.

1.3.5 RISK Cards

At the end of a player's turn, that player will receive a special "RISK card" to keep in their hand if and only if the player captured a new territory during their attack phase [2]. Each RISK card has a territory, and one of three symbols indicated on it, except two "wild cards" which are indicated as such. The symbols are arbitrary, but a standard version of the game uses the three pieces used to represent armies on the game board: a soldier, a cavalryman, and a cannon.

A player can remove a set of three cards from their hand if the set matches any one of three conditions. First, each card in the set contains the same symbol. Second, each card in the set contains a unique symbol. Lastly, at least one card in the set is indicated as a wild card. A player receives some number of bonus armies to place at the start of their next placement phase whenever that player removes a set of cards from their hand. Additionally, a player must remove at least one set of cards from their hand whenever that player holds five or more cards.

The number of armies that a player receives for removing a set of cards from their hand depends on the state of the game at that moment. The amount of bonus armies depends primarily on how many sets of cards have been removed from all players' hands. Initially, a player gets twice the number of sets removed, including the set being removed, plus two. The formulation changes after the fifth set, instead of using five times the number of sets removed, including the set being removed, then removing fifteen. Finally, two additional armies may be placed onto a specific territory indicated by any one RISK card in the removed set if the player removing the cards from their hand already has at least one army placed onto the territory. A breakdown for determining bonus armies appears in Equation 5. Variable n indicates the number of sets removed from the hands of all players, including the set currently being removed, and b indicates if a territory controlled by the player is shown on any of the cards in the removed set.

$$Armies = f(n, b) = \begin{cases} 2n + 2, & n < 6, b = FALSE \\ 2n + 4, & n < 6, b = TRUE \\ 5n - 15, & n \geq 6, b = FALSE \\ 5n - 13, & n \geq 6, b = TRUE \end{cases}$$

Equation 5. Equation for determining bonus army payouts when discarding RISK card sets.

1.3.6 Game Ending Conditions

Each player's goal in a game of RISK is to place at least one army onto each territory on the game board. Any player who achieves this goal is the winner of the game. Any player who lacks armies on the game board has lost the game, and the remaining players skip their turn until a different player wins the game.

1.3.7 Rule Modifications

I used a variant of the typical ruleset in RISK in this implementation. These rule changes simplify the implementation and reduce randomness between simulations.

The first rule change is the complete removal of RISK cards. While these cards add a layer of surprise and intrigue to human players, their primary purpose is to introduce randomness and reduce the importance of effective strategy by allowing surprise comebacks. Because the learning model depends on the repeated effectiveness of particular strategies, a mechanic specifically designed to undermine its effectiveness is problematic. The problems caused by the random nature of the RISK cards are reflected in their removal from other popular ruleset variants [2] [4].

The final change to the ruleset imposes a 100-turn limit on each player. The game ends when all players with armies still on the board reach their turn limit. A winner is chosen among players with armies still on the board upon reaching the turn limit using a new set of tie-breaking rules. RISK has the potential to drag out end game scenarios where one player has dominant control of the board but has a difficult time removing surviving players with only a few territories. This rule change keeps games relatively short, agent iterations frequent, and the simulation runtime quick.

1.3.7.1 Tie-Breaking Rules

Following the original ruleset in RISK, ties are impossible as the game only ends once one player has claimed every territory on the game board. However, with the introduction of the 100-turn limit outlined in the rule modifications, the chance of multiple players claiming territories on the game board is high.

The first step in determining the winner of a game in the event of a tie is to determine the total number of territories claimed by each player. The player with the most territories is the winner. If the first step did not resolve the tie, the second step is to determine the total number of

armies in play when the game concluded. The player with the most armies is the winner. If the second step did not resolve the tie, the third step is to determine how many armies a player would receive at the start of their next turn, assuming nothing else within the game state changed. The player that would receive the most armies at the start of their next turn is the winner. If a winner could not be determined after these three steps, the game is considered a proper tie, and all remaining players are winners.

1.4 Stochastic Beam Search

When it comes to AI agents and strategy games, there appear to be two camps: those that take the best action at the moment and those that plan ahead and take the action that will lead to the best outcome. The former is a powerful but simpler agent incapable of contingent strategies, while the latter is forward-thinking and reacts to the other players [1, pp. 120-189]. Typically, strategy games are better played by the more forward-thinking agents that have seen much success in games like tic-tac-toe, checkers, and chess [1, p. 190]. Unfortunately, as shown by Michael Wolf from Darmstadt University, the number of possible game states in a typical game of RISK is far too large to explore entirely [5]. Even partial glimpses forward require severe limitations on what actions are possible for an agent to undertake [5].

With the traditional approach to artificial intelligence for a strategy game environment made inaccessible, researchers must be creative in limiting the actions that an agent may take on a given turn or focusing on implementations that do not require speculating future game states. This research attempts the latter by using a stochastic beam search. Stochastic beam search tries to adjust input parameters to maximize the output of some function [1].

In this implementation, an artificially intelligent agent is composed of dozens of individually tuned characteristics. Each characteristic defines an agent's preference for how to react to a given situation. The stochastic beam searching algorithm adjusts these values as a method of input for a fitness function.

The fitness function in this implementation is a double-elimination tournament. Agents that outperform other agents in the tournament become the basis for future generations of agents. Each new generation of agents slightly adjusts the characteristic values from the top performers of the previous generation. Ideally, agents with characteristic values that produce effective strategies will perform well in the tournament, share those characteristics with other agents in the next generation, and slowly converge on a set of characteristic values that allow the agent to win at RISK consistently.

Chapter 2. Literature Review

The scientific inquiry into RISK appears limited despite its popularity and long history. However, that has not stopped enthused players from sharing their strategies that have led to anecdotal success.

2.1 Blogger Strategies

Blog posts are hardly steeped in scientific rigor, but they do provide some insight into common strategies within the RISK community. A common strategy proposed is to think in terms of continental control instead of territorial [4] [6]. Defending the borders of continents should be a high priority [4] [6] [7]. Similarly, capturing even a single territory within a continent should be a high priority if it robs an opponent of their continent bonus at the start of their next turn [4].

Common advice for new players is to play defensively [6] [7] [8]. Players should not rush into a fight and should usually try to avoid aggressive contact with other players whenever possible [6] [7] [9] [8]. Players should be sure to place many armies along the territories bordering a stronger player's territories to discourage attacks [4] [7] [8]. Although, there is some disagreement about this strategy because others feel a preemptive strike is typically a more appropriate response [6] [9].

The first territories that a player attempts to claim should be on either the Australian or the South American continents because they have limited connections to other continents and are therefore easier to defend [4] [6] [7] [8]. It is always better to attack with a large number of armies relative to the defender rather than an equal number of armies [4] [7] [8] [9].

When defending, it is always preferable to use two dice in a battle [4] [9]. Likewise, an attacker should always use three dice in a battle when possible. Especially if the defender only uses a single die [4] [9].

2.2 Attack Strategies

Attacks between territories are complex interactions between combinations of random numbers, specifically sets of dice rolls. While there has not yet been a rigorous review of the statistics of these outcomes, many enthusiasts of the game casually explore these probabilities.

For example, Nick Berry from Data Genetics wanted to examine the odds of an attacker winning a battle between two territories [10]. Berry reasoned that the most dice a single battle during an attack in RISK uses is five—three dice from the attacker and two from the defender. From this, he deduced that five dice have at most 7,776 distinct combinations [10]. Realizing that this upper limit is well within the bounds of what contemporary computers are capable of processing, he calculated every possible combination of dice to determine the results.

The results were interesting in that they showed a consistent advantage for attackers. Of the six possible combinations of attackers and defenders, four cause the defender to lose at least one army [10]. Somewhat contradictorily, when defenders have an advantage, they have comparatively more advantage. For example, in the best-case scenario for an attacker, three attacking armies versus one defending army, the attacker only has a 66% chance of winning the battle. These odds are significantly lower than the best-case scenario for a defender, i.e., one attacking army versus two defending armies. In the best-case scenario for a defender, the defender has a 75% chance of winning the battle.

Berry used the probabilities of each change in attackers and defenders as the result of a battle to create a tree-like graph to model a series of battles in an attack [10]. Each node in the graph correlates to some state of attackers and defenders remaining. Each edge connecting nodes represents the probability of transitioning to that state after a battle. By tracking what percentages of nodes eventually lead to zero defenders, he calculated the odds of an attacker successfully destroying all armies on the defending territory.

Iain Hart from the University of Manchester developed a similar table of probabilities [11]. However, unlike Berry's implementation, Hart's relies on the law of large numbers to converge onto the real statistics. His results are nearly as accurate as Berry's, except the implementation is simpler and computes faster. Hart's table shows probabilities of a successful attack, i.e., all defenders destroyed, and successful defense, i.e., all attackers destroyed. His table calculates values up to 29 attackers against 29 defenders. The probability of a successful defense is the complimentary percentage of a successful attack. For example, if an attacker has a 25% chance of success, then the defender has a 75% chance of success.

Garrett Robinson from the Massachusetts Institute of Technology used a Monto-Carlo simulation to calculate the chance of a successful attack between two groups of armies [8]. Given the number of armies in each group, the Monte-Carlo simulation would randomly pick the number of attackers and defenders with whom to battle. The battles would continue until either the attackers or the defenders were eliminated. Robinson simulated attacks with up to 30 attackers and up to 30 defenders, then simulated each permutation 100,000 times [8]. By tracking the number of successful attacks, the Monte-Carlo method converges on the expected probability of a successful attack given the sizes of the two groups. Unfortunately, Robinson's research

made no commentary about the strategy of using different combinations when battling. However, it did support the conclusion that whichever group has more armies is more likely to win [8].

2.3 Wolf's Implementation of Artificial Intelligent RISK Agent

Michael Wolf from the Darmstadt University of Technology detailed his implementation of an artificially intelligent agent capable of playing RISK in 2005 [5]. Wolf's implementation is based around a serializable game state map representing the game board's map, a rules object that provides all possible actions for a player for a given game state, and an interface for an agent to decide what actions to take upon consideration of the game state. When considering potential attacks, the estimates are generated using a method similar to the one described by Berry from Data Genetics [10] [5].

Wolf actually developed three artificially intelligent agents: one with random behavior, one with basic behavior, and one with more advanced behavior [5]. The random-behavior agent randomly selects actions when playing and is the least effective of the three behaviors but did establish a baseline level of performance that was useful for analyzing the other behaviors.

2.3.1 Basic Behavior Agent

The basic behavior agent uses a series of evaluation functions to determine the desirability of a game state resulting from taking a specific action. The desirable traits of each game state are hand crafted by Wolf. Whichever legal action leads to the best-evaluated game state will be the action taken by the player agent. When multiple actions lead to the same game state or produce game states of equal value, the agent randomly selects an action to take.

2.3.2 Advanced Behavior Agent

The advanced behavior agent attempts to compensate for the basic behavior's perceived shortcomings. Namely, it seeks to correct the basic behavior's inability to coordinate evaluation functions between different phases of a player's turn and its inability to create multi-turn strategies. The behavior includes the same system of evaluation functions used by the basic behavior but includes two new high-level evaluations and one new type of action. The first new consideration is how a given action affects the game state in a specific continent of interest, or "target continent" [5]. Another new consideration is how a given action moves the game state to some pre-defined and high-level short-term goal. The new type of action allows armies to be placed onto territory in batches instead of placing each army individually. Lastly, a learning algorithm adjusted the weighted preference of the now competing considerations.

Each turn, a new target continent is selected so the agent can adapt to drastic changes in the game state and adjust its strategy accordingly. The agent will prefer game states that benefit a particular continent. For example, an agent will prefer game states that place more armies within that continent's territories, moves armies from other continents into the target continent, and allow the agent to claim additional territories within the continent. This change allowed the agent to concentrate armies and activity on specific continents. Consequently, that behavior change led to better play because the agent could conquer continents more quickly and keep them better defended when under their control.

Plans seek to remedy the agent's inability to strategize across multiple turns by marking arbitrary territories as preferable targets for attack. For example, one plan may prioritize attacks

on territories controlled by a specific enemy player. Alternatively, a plan may prioritize all territories on a specific continent.

The new action tries to coordinate efforts between the placement evaluation and the attack evaluation functions. The agent with the described basic behavior reportedly had an issue with spreading armies too thinly across claimed territories. This issue prevented the agent from properly defending borders. Additionally, the agent would often not have enough armies adjacent to an enemy claimed territory to capture the territory in an attack successfully, but otherwise would have been able to if the agent had placed more armies onto the attacking territory during the placement phase. Placing armies in batches equal to some percentage of the total number of newly available armies helped avoid the behavior described by the agent using the basic behavior.

Wolf used temporal difference learning to dynamically adjust the weighted preference of each consideration: the basic behavior's choice, the target continent, and the plan [5]. After completing each player's turn, the agent can determine how much the game state evaluation has changed from now compared to the end of their last turn. Depending on much the game state has decreased, the agent can learn to prioritize actions according to the preferences of the basic behavior, the target continent, and the high-level plan.

2.3.3 Conclusions

Wolf concluded that agents using the advanced behavior significantly outperformed agents using the basic behavior across 1000 games [5]. Both the advanced and basic behavior agents dominated the agent using random behavior. Wolf additionally concluded that the inclusion of plans had the most significant impact on an agent's performance. Additionally, the

agent learned to heavily favor actions beneficial to the current plan. Wolf's suggestion for future research was to increase an agent's ability to learn and develop a method for dynamically creating plans beyond what a program could pre-define.

2.4 Erik Blomqvist's Implementation of an AlphaZero Agent

Erik Blomqvist of the Kth Royal Institute of Technology created an artificially intelligent agent that played RISK by combining a Monte Carlo Tree Search algorithm with a neural network that learned how to evaluate game states over time [12]. The agent is limited to only games with two players, plus a non-attacking neutral third player recommended by the original ruleset [2] [12]. Additionally, while the ruleset allows for cards, the players must play the cards immediately when possible. Furthermore, cards always give the same number of armies upon trade-in. The other changes to the base ruleset affect how attacks are processed. When attacking, a player must use all available armies to attack, attacks cannot be canceled partway through, and the attacker must roll the maximum number of dice available. Likewise, a defending player must always roll the maximum number of dice available when defending.

Each turn, an agent simulates taking a collection of random actions, then simulates its opponent taking its own set of random actions. Once the simulation has looked sufficiently far ahead, the hypothetical game state receives a fitness score. The fitness score backpropagates up to the original simulation, where the actions taken are then recorded and stored with the resulting fitness score. To account for the stochastic nature of attacks, each attack split the search tree and forced the exploration of all potential outcomes. During backpropagation, an attack node's fitness score is the sum of all back-propagating scores multiplied by the probability of that branch occurring.

The neural network trains itself using an algorithm called EXIT that is itself based on the more popular, but difficult to implement, AlphaZero algorithm. The algorithm is strange because it trains itself without outside influence or reinforcement.

2.4.1 Conclusions

The paper was more focused on advancing and promoting the EXIT and AlphaZero algorithms than on improving strategic play at RISK. The researchers did conclude that the learning agent outperformed a different agent using hand-crafted state scoring in 144 of 200 matchups [12]. Blomqvist noted that the agent failed to reach any superhuman level of play at RISK like what has been seen by applying the same algorithms to other board games [12].

2.5 Stochastic Beam Search

Stochastic beam search is a popular method for solving optimization problems [1]. The algorithm randomly adjusts many different combinations of inputs to maximize the output of some function. After evaluating a collection of combinations, a random selection of some of the most performant are modified to create a new collection. The new collection has each input combination evaluated just like the first collection. This process repeats an arbitrary number of times or until the search space is exhausted. The collection slowly converges on more optimal combinations because underperforming combinations are actively removed from the collection to accommodate the top-performing.

Thinking of each combination of inputs as a collection of genes in an organism is a common and helpful way to conceptualize stochastic beam search [1]. Extending the metaphor further, the collection of input combinations becomes a population of organisms, and the function to maximize becomes the environment in which the organisms must survive. Each

iteration is then synonymous with the lifecycle of the origin. Organisms that fail to perform as well in their environment die out, i.e., are removed from the collection. The organisms that do perform well reproduce, i.e., are duplicated across the collection.

Stochastic beam search is closely related to the more popular genetic search algorithm, except it lacks the crossover method described in genetic algorithms [1]. During crossover, a genetic algorithm selects two parent organisms to copy long sequences of genes over from each parent into the offspring. Continuing the organism metaphor, genetic search is comparable to sexual reproduction, and stochastic beam search is comparable to asexual reproduction. While merging long sequences of inputs can be beneficial in some specific implementations, genetic search and stochastic beam search have been shown to produce similar results [1] [13].

Chapter 3. Approach

3.1 High-Level Overview

Each artificially intelligent agent exists within a population of other artificially intelligent agents. Each one is likely unique from one another according to their agent definition. An agent definition is the set of characteristics that describe how the agent behaves in a given situation. Each agent competes in a double-elimination tournament, with the top 25% becoming the basis for the next generation of agents. Each new generation is a randomly altered population of the previous generation's winners.

For each phase of a player's move, an agent considers the game board via a well-defined map of the territories and continents. The player considers the impact of each possible placement, the estimated chance of attack of each attack, and the impact of each army moved. As each decision is made, the agents acts on behalf of a player and coordinates the appropriate actions take place upon a map. Each game of RISK in the tournament limits a player to only 100 moves, so agents must conquer territories quickly to achieve victory.

Across many tournaments, the agents will slowly adapt to defeat their opponents. Agents who fail to defeat their enemies are removed from the population, leaving only those agents capable of good strategic decision making until they are all that is left. This approach, known as stochastic beam search, will eventually converge onto a powerful agent definition capable of playing RISK at a competitive level.

3.2 Agents

The backbone of the project's implementation is the Agent class. This class is used to manage and interact with a collection of Agent Characteristics. The heart of the data structure is

an associative array of associative arrays. The underlying associative arrays map characteristic names to Agent Characteristic objects.

3.2.1 Agent Characteristics

The first layer of the associative array identifies a category of characteristic. This logical distinction allows the object to be more easily extended by appending new characteristics to the appropriate group, or by appending an entirely new set of characteristics into a new category. Another benefit is using multiple characteristic names across multiple categories without worrying about key collisions. Collectively, this layer comprises the Agent Definition because it defines the core behaviors that an agent will use when making decisions.

The second layer of the associative array identifies individual Agent Characteristic objects. An Agent Characteristic is a low-level data structure within the project that stores basic information about an agent’s preferences and considerations.

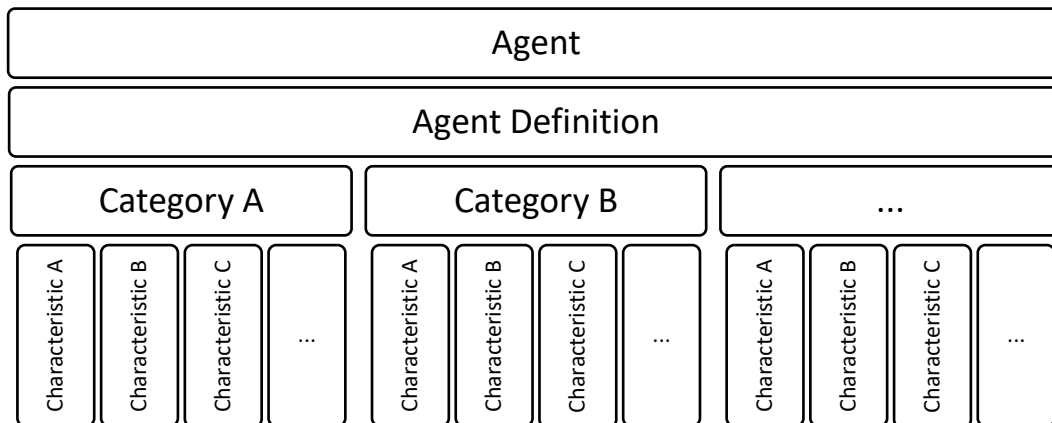


Figure 3. Graphical interpretation of Agent data structure.

Each Agent Characteristic contains a numerical value, an upper and lower bound for the value, an increment value, and a high-level human-readable description of what the Agent Characteristic is supposed to represent. The increment value is used to adjust the numerical value

of the object either positively, negatively, or randomly. The description is useful for debugging, storing display information, and meta-processing after serializing to a common format such as JSON, XML, or YAML. The numerical value can be used to represent anything as long as the bounds and increment amount are adjusted accordingly.

Each Agent Characteristic within this second layer is mapped to a unique name used to identify and reference the object elsewhere within the program. Ideally, the name should be human-readable and should coordinate with the Agent Characteristic's description of itself.

3.2.2 Agent Mutation

An Agent can mutate its Agent Characteristics over time. An agent can mutate its Agent Characteristics in three ways: major, minor, and cascading mutations. A minor mutation selects a single Agent Characteristic within a single category of Agent Characteristics, then randomly adjusts the value. A major mutation selects a single category of Agent Characteristics, then randomly adjusts the value for each Agent Characteristic in the category. A cascading mutation randomly selections between major and minor mutations, with an additional random chance of recursively causing another cascading mutation. Each time a cascading mutation causes another cascading mutation, the chance of causing another cascading mutation is halved. By default, a cascading mutation initially has an 80% chance of causing another cascading mutation. For example, the chance of a cascading mutation occurring three times is 6.4%.

3.3 RISK Agents

To create an Agent Definition specific to RISK, I first divided the characteristics into six main categories: placement, attack, movement, preference, attack continent, and place continent. The placement category is concerned with how the agent decides onto which territories to place

armies. The attack category is concerned with how the agent determines when to attack, which territories to attack, and which armies to use. The movement category is concerned with how the agent determines to move armies from one territory to another during the movement phase of their turn. The preference category is the most loosely defined. This category concerns itself with high-level behaviors of an agent, e.g., tolerance for tactical risk. The attack continent and placement continent categories provide additional flexibility in planning by favoring particular territories in particular continents when attacking and placing armies, respectively. A full breakdown of the agent definition can be found in Appendix A: Agent Definition.

3.3.1 Placement Phase

A RISK Agent defines a method for placing armies onto the game board. First, the agent requests a collection of all territories under its control from a given Map object. The agent uses a separate method to pick a single territory from the collection that will receive a new army. The agent then requests the Map object to place a new army on that territory. The driver for a game of RISK may call this method for each new army that the agent may need to place during the placement phase of their turn. The same process is used for the initial setup of the game board, except the agent first picks from a collection of territories not claimed by any players until all territories have been claimed.

The method responsible for picking a territory is complex and references every agent characteristic in the placement and preference categories. Each territory in a provided collection of territories is scored, then the territory with the best score is returned. If no territory is determined to be the best, then a random territory is returned. If multiple territories have the same score, then the first territory in the collection that produced that score is returned.

To score a territory, the agent first scores the territory according to the characteristics in the “Placement” category. First, the score is set to the value defined by the “Anywhere” characteristic. Next, the territory is adjacent to enemy territory, i.e., a territory claimed by another agent, then adding the value in the corresponding characteristic to the score if necessary. A similar process is used to adjust the score if the territory is adjacent to a territory also claimed by this agent, and if the territory is adjacent to a territory along the border of a continent. Each of these checks trivially gets the appropriate data from the connected Territory objects. Afterward, the agent adds a “Connection Bias” value to the score for each territory that the current Territory object is connected to.

After the score is adjusted by the characteristics in the “Placement” category, the score is then adjusted according to the “Preferences” category. If the territory is not adjacent to an enemy territory, then the value of the “Safe” characteristic is added to the score. If the territory is adjacent to an enemy territory, then the current territory is compared with the current best territory.

Comparing the current territory with the current best territory allows the agent to take calculated risk and target players with either more or less control over the game board. First, the connections of both Territory objects are reviewed to determine the territory with the largest army controlled by another player. If the current territory is connected to an enemy territory with a larger army than the current best territory, then this territory is considered to be riskier option and the value of the “Risky” characteristic is added to the score.

Afterward comparing the relative risk of placing a unit on the current territory with the current best territory, the agent considers the army placement from a high-level perspective. The

connected Territory objects of both territories are reviewed again to determine which territory is connected to enemy territory with the most influence over the game board. First, the agent considers the size of each player controlling a territory connected to the current territory. Next, the agent repeats this process for the current best territory. If the assessed size of the largest player connected to the current territory, according to the Map Class, is larger than that of the current best territory, then the “Larger” characteristic value is added to the score. Similarly, the “Smaller” characteristic value is added to the score if necessary. If both territories are connected to the same largest player, or the players have the same size, then neither value is added to the score.

The last step of the scoring process introduces diminishing returns for placing armies onto territories that already contain many armies. This step prevents agents from simply placing all armies onto the same territory because that territory scored the best at every check. To calculate the rate of this diminishing return, the “Placement Bias Multiplier” value is raised to a power equal to the number of armies currently on the territory. The current territory’s score is then multiplied by that value. For example, if the “Placement Bias Multiplier” had a value of 0.9, and the current territory already had five armies placed upon it, then the score would be multiplied by approximately 0.59. Lastly, if the final result of the current territories score is higher than the result of the current best territory’s score, then the current territory becomes the new current best territory.

3.3.2 Attack Phase

3.3.2.1 Attack Preparation

A Risk Agent carefully considers what attack options are available and how taking control of a new territory might have far-reaching effects across the game board. Each possible attack is summarized as a final attack impact score. This score is how an agent will make its ultimate decision about what territories it will attack, what territories will provide the armies, and if any territory is worth attacking at all.

However, before an agent can ponder these game-changing repercussions, it must first know what territories it can attack. A Map Object can provide a collection of all territories claimed by a player. This collection is filtered down to just those territories with more than one army on the territory, as it takes two armies to initiate an attack on another territory. The collection may be reduced further to only those territories that connect to at least one other territory not claimed by the same player. However, I chose not to do this in my implementation as the situation is uncommon until the game is nearly over, and the connected territories will be iterated over again momentarily.

Now that the collection of potential attackers has been considered, an agent can start to consider the outcome of each possible attack. At this step, the agent starts iterating through each territory in the collection of possible attackers and iterating through each of their connections to territories claimed by a different player.

3.3.2.2 Attack Estimates

The first step of many when determining if an attack between two territories should occur is to request an attack estimate from an Attack System object. If the attack's chance of succeeding is less than the agent's "Minimal Success Chance" characteristic from the "Attack" category, then the agent does not consider this a viable attack, and immediately starts to consider other options.

Additionally, if an attack's chance of succeeding is lower than the value of the "Safe Threshold" characteristic from the "Attack" category, then the attack's impact score is increased by the value of the "Safe" characteristic from the "Preference" category. Inversely, suppose the attack's chance of succeeding is higher than the value of the "Safe Threshold" characteristic. In that case, the impact score increases by the value of the "Risky" characteristic from the "Preference" category.

Next, it will consider two other values returned from the estimate, how many attackers are expected to survive and how many defenders are expected to survive. The estimated number of surviving attackers is multiplied by the "Survivorship Bias" from the "Attack" category. The estimated number of surviving defenders is first subtracted from the total number of armies currently on the territory to transform the estimate into an expected number of defenders destroyed. The newly calculated estimate of defenders destroyed is multiplied by the "Targets Destroyed Bias" characteristic from the Attack Category. Both of these products are added to the attack's impact score.

3.3.2.3 Attack Impact

The attack's impact score is then affected by much more straight forward measures. We can next add the "Anywhere" value from the "Attack" category. This is awarded to any attack and is included so that an agent may adjust how practical it is to attack anywhere. The value could be negative if the agent is more defensively inclined and it does not want to attack anywhere when not especially valuable to do so. Alternatively, the value could be positive to promote aggressiveness and force the agent to attack unless it is especially prudent not to attack.

The attack's impact score is adjusted by the "Ally Adjacent" characteristic from the "Attack" category if the defending territory is adjacent to at least one other territory controlled by the attacking player. This can be accomplished trivially by iterating over the defending Territory object's connections.

The next variable to influence the attack's impact score is if the defending territory is adjacent to another continent. These territories define the border of a continent which is why the characteristic value added is from the "Border Adjacent" characteristic from the "Attack" category.

An agent adds the "Capture Continent" value from the "Attack" category if and only if capturing the territory would mean the attacker claiming control over every territory in a continent. The easiest, although certainly not the quickest method, to do this is to request the number of armies received from a continent bonus via a Map object then store the result. Next, change the owner of the defending territory to the attacking agent and request the continent bonus from a Map object again. If the value has changed, it can be deduced that the territory was captured. If the value is the same, then capturing the territory will not allow the agent to claim

every territory within a continent. If this implementation is used, one must take special care to restore the defending territory to its previous owner before proceeding.

Lastly, it will consider the size of the defending player. The size is determined by a Map object that quantifies a player's size by how many territories they claim and how many armies they own. When considering multiple attacks, it is often good to compare how a current option weighs in comparison to the best-known option so far. If the defending player is comparatively larger than the defending player of the best attack option found so far, then this agent may have a preference for attacking larger players and will add the value of the "Larger" characteristic to the attack's impact score. Inversely, if the current defending player is smaller than the defending player in the best attack considered so far, then the "Smaller" characteristic's value is added to the attack's impact score. Both the "Larger" and "Smaller" characteristics are defined in the "Preference" category.

Once all possible attacks have been iterated over, and the attack with the highest impact score has been determined, there is one final check. Occasionally, it is better to refrain from attacking in RISK if it will put you into a worse position strategically. A RISK agent can decide if the best attack available is a worse move than simply not attacking anymore, if at all this turn. To make this decision, the best score is compared to a "Minimum Impact Score" characteristic from the "Attack" category. If the best score is larger than the value of the "Minimum Impact Score," then the agent has decided to actually perform the attack. If the best score is smaller than the "Minimum Impact Score" value, then the agent has decided that its best move is less preferable than not attacking and will therefore not attempt to attack.

3.3.3 Movement Phase

How a Risk Agent chooses which armies to move to and from where is a deeply complex process that is possibly more strenuous than choosing what and how to attack territories. Many steps in the operation are similar to those taken when attacking. At a high level, the process involves filtering out territories by those eligible to move armies, giving each possible an impact score, contrasting its impact to the currently best-known movement, then deciding how many armies to move, if any.

An agent can begin the movement considerations process by first filtering all claimed territories by the agent to just those capable of supplying armies for movement, i.e., those with more than one army. The territories capable of receiving armies are therefore all territories connected to at least territory capable of supplying armies and is claimed by the same agent. Together, an army supplying territory and an army receiving territory make up the beginnings of a well-defined movement. All that remains are several armies to transfer between the territories and an impact score.

3.3.3.1 Quantifying Risk of Movement

First, the agent determines if a movement is either risky or safe. A risky movement is any movement that removes armies from a territory connected to at least one territory claimed by another player. By contrast, a safe movement is any movement that removes armies from a territory not connected to any territories claimed by another player. Examples of how movements are classified as a safe or risky can be seen in Figure 4.

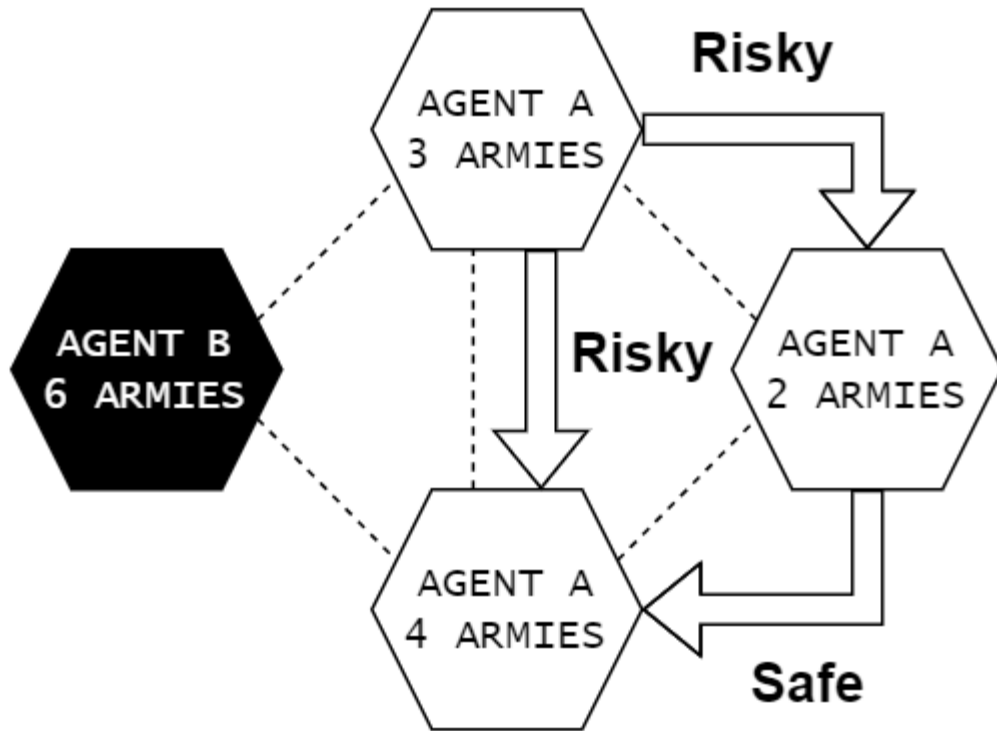


Figure 4. Examples showing how movements may be classified as either "Risky" or "Safe" behavior.

Once a movement is known as either risky or safe, the agent can decide about how many armies to transfer between the territories. The percent of units to transfer are determined by the "Risky Transfer Rate" and "Safe Transfer Rate" from the "Movement" category as appropriate. The actual number of armies to transfer equals the number of armies on the territory multiplied by the percent transfer rate, then rounded down to the nearest whole number.

If the number of armies to transfer ever request more armies than are eligible to be moved, then the transfer amount will instead be the number of armies on the territory minus one, because at least one army must remain on the territory. Therefore, we derive Equation 5 for calculating the number of armies to transfer.

$$Armies\ to\ Transfer = \begin{cases} \left\lfloor \frac{Transfer}{Percent} \times \frac{Armies\ on}{Territory} \right\rfloor, & \left\lfloor \frac{Transfer}{Percent} \times \frac{Armies\ on}{Territory} \right\rfloor < \frac{Armies\ on}{Territory} \\ \frac{Armies\ on}{Territory} - 1, & \left\lfloor \frac{Transfer}{Percent} \times \frac{Armies\ on}{Territory} \right\rfloor \geq \frac{Armies\ on}{Territory} \end{cases}$$

Equation 6. Equation for determining how many armies to move between two territories given a transfer percent.

Now that the agent knows how many armies to consider moving to and from what territories, it can begin to calculate the movement's impact score. The impact score initiates to the value of either the "Risky" or "Safe" characteristic from the "Preference" category as appropriate.

3.3.3.2 Movement Impact

Next, we can add the "Anywhere" value from the "Attack" Movement. This is awarded to any movement and is included so that an agent may adjust how practical it is to move anywhere. A highly positive value could promote agility, while a low or negative value may suggest that moving is an unnecessary strategy and that the agent relies exclusively on the army placement phase to get armies where they need to be.

The impact score is affected by the "Enemy Adjacent," "Ally Adjacent," and "Border Adjacent" characteristics from the "Movement" category by trivially iterating over the connections of the receiving territory using the same implementations described when attacking.

The impact score is next adjusted based on the number of connections to other territories. The number of connections to the receiving territory is multiplied by the value of the "Connection Bias" characteristic, then the product is added to the impact score.

Additionally, the number of armies on the supplying and receiving territories also affects the movement's impact score. If the receiving territory has more armies than the supplying

territory, then the “Bigger Territory” characteristic is added to the score. Otherwise, the “Smaller Territory” characteristic is added to the impact score instead. Both the “Bigger Territory” and “Smaller Territory” characteristics are defined in the “Movement” category of the agent definition.

The final adjustment to the movement’s impact score occurs when comparing the current movement under consideration with the best movement considered so far. First, the agent considers all territories connected to the receiving territory and what players own those territories. This step occurs for both the current and the best-known receiving territory. Assuming that an enemy connection exists for both territories, the size of the largest player connected to each territory is considered. To clarify, the player’s size is a single magnitude representation of how powerful a player on the game board is, as determined by a Map object.

If the receiving territory of the best-known movement is connected to a territory claimed by a larger enemy than the current receiving territory, then the “Preference” category’s “Smaller” characteristic’s value is added to the impact score. If the opposite is true, and the best-known movement has its receiver connected to a territory claimed by a smaller enemy, then the “Larger” characteristic’s value is used instead. To clarify a point on the naming convention of these characteristics, the “Smaller” and “Larger” refer to the size of the current consideration in comparison to the currently best-known movement considered.

Finally, after all movements have been considered, a final check occurs to determine if it is even worth moving any armies at all this turn. The best-known movement is compared to the “Minimum Impact Score” characteristic from the “Movement” category. If the best-known impact score is less than the “Minimum Impact Score,” then the agent has decided that it is not

worth moving any armies between territories this turn. Otherwise, the agent will commit to moving.

3.4 Attack System

Attacking and defending in RISK is a complex process that introduces stochastic elements to the game to make it more enjoyable for human players. Recall that attacking in RISK relies on the results of a series of dice rolls. Each attack is conceptualized as a series of battles with the attacker using either one, two, or three dice, while the defender uses either one or two dice. Each die is correlated with an army attacking or defending territory. While each agent processes the ability to initiate an attack so that it can make fine-tuned adjustments depending on its characteristics, the process of attacking is handed over to a Map Object, which in turn defers the calculations of attacking over to an Attack System object.

The Attack System class outlines the behaviors necessary to conduct an Attack in RISK. These behaviors define how to roll virtual dice, conduct battles between armies, conduct attacks between territories, and provide estimates for the outcomes of attacks. First, this thesis addresses how an Attack System conducts an attack between territories from a bottom-up point of view, then explains the process for providing expected outcomes for agents to make informed decisions.

3.4.1 Simulating Attacks

The first responsibility of an Attack System is to simulate the roll of a die capable of generating a random number from 1 to 6. From here, the Attack System provides a method of retrieving a collection of a specific size where each element is the result of a random dice roll.

These new collections of dice rolls can be used to conduct battles. First, the attacker requests a collection of dice rolls. The attacking agent can request up to 3 dice, and the defending agent can request up to 2 dice. The requested size of the collection is determined by the “Attack Dice Count” characteristic from the “Attack” category. If the territory does not have the requested number of armies available, it will simply use the most available up to the requested amount. A similar process occurs for the defending agent, except the requested collection size is determined by the “Defend Dice Count” characteristic. After both collections have been created and populated, they are both sorted. While both collections have at least one element, the first element of each is removed and compared. When comparing elements of each collection, if the element from the attacker’s collection is smaller, it is noted that the attacker has lost an army. Otherwise, the defender has lost an army. The change army counts for both the attacker and defender are returned as a tuple result.

Now that the Attack System class has a method for conducting battles, that method can be used to conduct attacks. The attack method calls the battle method in a loop until the attack has ended, tracking the army count of each territory after each battle. An attack ends once the defender has no more armies to defend with, the attacker has no more armies to attack with, or the attacker has lost the highest amount of armies it is willing to on this attack. This method of ending an attack is adjusted by the characteristics of the agent initiating the attack. The Attack System class’s attack method returns the final amount of armies on both the attacking and defending territories as a tuple result.

3.5 Using Estimates to Make Predictions

The Attack System class is responsible for all activities related to combat, including estimates. Attack estimates are crucial for the Risk Agent class as it allows it to consider the possibility of success and failure of attacking a territory.

3.5.1.1 Simple Estimates

Before any game of RISK begins, tables of data are pre-computed that an Attack System object can reference during a game. To generate these tables, an Attack System object simulates 5000 attacks between territories of every size from 1 to 30. The simulation tracks the average chance of the attacker successfully destroying every defending army, the average chance of the attacker successfully destroying every attacking army, the expected number of surviving armies when the attacker wins, and the expected number of surviving armies when the defender wins. This simulation process for calculating attack statistics is similar to the method used by Iain Hart [11].

When a Risk Agent requests an attack estimate from an Attack System object, the agent will provide the number of available attacking armies and the number of defending armies. If the number of both armies is less than 30, then the Attack System will return the precomputed attack statistics.

3.5.1.2 Elaborate Estimates

If either the amount of attackers or defenders is more than 30, then the numbers are converted into a ratio. The Attack System object then checks for a precomputed attack statistic

with the same ratio of attackers to defenders. If the precomputed ratio exists, the estimate returns the success chance of the precomputed ratio.

If the precomputed ratio does not exist, the attack system attempts to adjust the ratio by subtracting one from both numbers in the current ratio until an existing ratio is found or either value reaches zero. If an existing ratio cannot be found before either the number of attackers or defenders goes to zero, the estimate of success is set to 100%, and the estimated number of remaining armies is simply the remaining armies after so many steps of the subtraction process. The number of remaining armies is estimated by scaling up the estimated value in the precomputed ratio.

The method whereby each value in a ratio is subtracted by one works surprisingly well for this purpose. For example, consider that an attacking territory has 86 armies, and the defending territory has 17 armies. This 86:17 ratio will not reduce to something with both values less than 30, so the ratio is changed to 85:16, which will not reduce. The process repeats, changing the 85:16 ratio to 84:15, which can reduce to 28:5. Because both values are now less than 30, statistics for that ratio already exist. The estimated number of surviving attackers in the found ratio is 23. To find the estimated number of survivors for the requested ratio in the example, multiply 86 by $23/28$ to get a little more than 70 surviving armies. This implementation is pessimistic and always rounds down to the nearest whole number of surviving armies.

3.6 Map

The Map class mediates all interactions with the state of the game board. The underlying data structures are a pair of associative arrays: one for territories and the other for continents. Figure 5 is a visual representation of the Map data structure. It shows how the map object is built

up out of two collections—territories and continents—and how the territories have pointers to their respective continent (Figure 5).

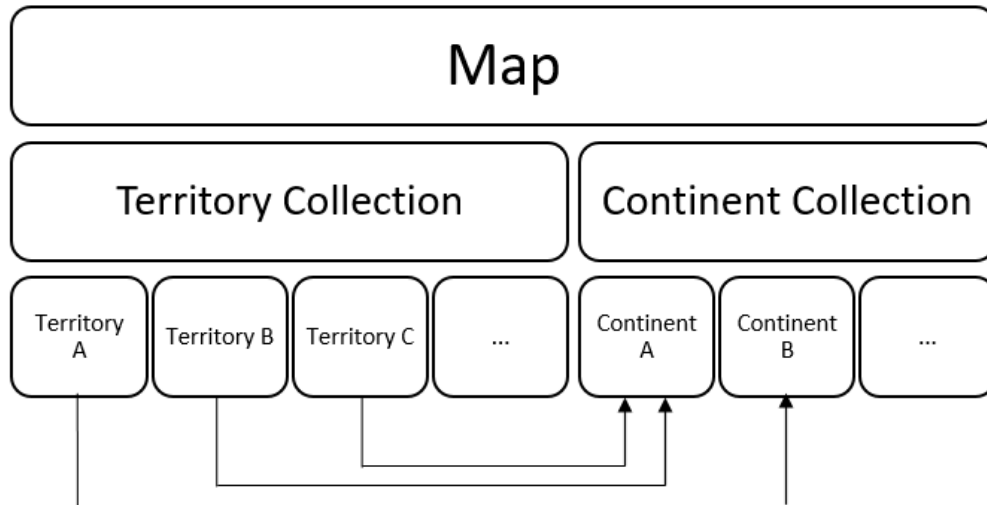


Figure 5. Graphical interpretation of Map data structure.

3.6.1 Territories

Recall that territories are the smallest measure of space on the game board that a player may place armies upon, and continents are sets of territories that provide bonuses to a player with armies on all territories within the set. The key for each territory is the territory’s index, and the continent’s key is a human-readable string of the continent’s name.

Each Territory object begins with a unique identifying index. The uniqueness is enforced by the Map object. Additionally, each Territory object stores a string representation of the continent they belong to, how many armies are currently on the territory, a string representation of the player in control of those armies, and a collection of indices used by other Territory objects to which it is connected. The Map class ensures that all connections are bidirectional. Lastly, Territories store a tuple of XY coordinates for the graphics renderer to reference when

drawing a graphical representation of the map. A Territory's index, connections, continent, and coordinates can be considered read-only data. The Map object ensures that a player's string representation stored in a Territory object is valid. The Territory class provides methods for adjusting the number of armies currently on the territory and ensures that a territory cannot have less than one army at any time.

3.6.2 Continents

Compared to the Territory class, the Continent class is much more simplistic. Each Continent object holds its own name as a human-readable string, a unit bonus as an integer value, and a color value for the graphics renderer to reference. The color provides a way to quickly distinguish territories by continent when viewing a graphical representation of the game board. A Continent object does not store a collection of territories because that data should already be mirrored and verified as accurate by the Map object inside of Territory objects.

The Map class provides various ways to pull information from the Territory and Continent collections it manages. A Map can determine if a player has armies on a given territory. From that, a Map object can produce a collection of all territories controlled by a given player by iterating over all Territory objects within its collection. With that information, it is trivial to produce the total of all armies on the game board controlled by a given player.

To produce a collection of all territories within a given continent, one can simply iterate through all Territories within the stored collection, categorizing each into an associative array with the continent as the key, and a collection of Territory objects as the value. It is now trivial to determine if all Territory objects within a Continent possess armies placed by a given player. Additionally, it is simple to determine if a player should receive bonus armies from a given

continent and how many armies it should receive. By performing this check across all continents, it is simple to deduce the total number of bonus armies a player should receive including all continental bonuses. Next, a Map object can calculate the total number of armies that a player should receive at the start of their turn using Equation 4. Complex base army count equation.

3.6.3 Maps as a Game State

Lastly, a Map object can recreate some in-game actions such as placing an army onto a Territory, attacking one territory from another, and moving armies between territories. To accomplish the simple in-game actions of placing and moving armies, a Map object simply adds or removes armies from a given Territory object. Simulating one territory attacking an adjacent territory is a more in-depth process that follows the rules established by an Attack System object, then adjust the territory objects' army count and controlling player's names accordingly. For each in-game action, the Map class ensures that each requested action is legal.

3.6.4 Serializing Map Data

Map data—the Continent and Territory data—is stored in a single file as serialized JSON data. The Map Reader class works as a Map factory. The most important method that the Map Reader class has in the *read map* method that accepts a file path as input and returns a newly instantiated Map object. First, this method opens the file at the given path and verifies the file exists and contains valid JSON data. Next, it begins to load and process the data in the file.

The first data collection within the JSON file is agent data, where Agent objects may be serialized and stored. This is useful for loading in Map data with specific game states in mind beyond the initial pregame state. The second collection of data within the JSON file is continent data. As this data is read, it is passed through basic validation checks that ensure that there is no

missing data, and that no data is illogical such as a negative army bonus. The third data collection within the JSON file is territory data that also passes basic validation checks as it is read. The basic checks ensure that all data is missing, that no territory data with the same index has already been processed, and that no data is illogical such as a negative number of armies placed onto the Territory. Furthermore, the checks ensure that if a set of territory data has a player's name included, that the player exists in the previously processed agent data. Similarly, a different check ensures that the continent name provided exists within the previously processed continent data.

Once all the Territory data has been processed for basic validation, a second round of validation begins. This second round of validation ensures that all connections between Territory objects are bidirectional. Once the interconnectivity between Territory objects is validated, the final Map object is instantiated with all properties necessary being set and all internal collections being populated. The final step returns the new Map object to the method caller.

3.7 A Game of RISK

The Risk Game class is a type of strategy game that coordinates in-game actions between the agents and the map to simulate a complete game of RISK. At a high level, the Risk Game class only has two methods: setup the game board, and play a game. While setting up a game board is fairly straightforward, playing a game of RISK is a complex event. To play a game, a Risk Game object needs to set up the game board, iterate through each phase of a player's turn, check if any end-game conditions have been met, and return the names of the winners and losers as a pair of collections.

3.7.1 Game Setup

Setting up the game board is a simple process to implement. First, given the initial number of armies each player receives, multiply that number by the number of players to get the total number of initial armies for all players. A quick check ensures that there are at least enough initial armies to place at least one army on each territory. Next, for each army in the previously calculated total, have exactly one agent place one army using the appropriate method. Before the next iteration, select the next agent to place an army. The process repeats until all initial armies have been placed on the game board.

3.7.2 Game Play

The game is now ready to enter its core gameplay loop, where each agent makes decisions as a RISK player for all phases of their turn until an end-game condition is triggered. First, the Risk Game object requests the new army placement count for the current player. Next, the agent, i.e., the current player, places that number of new armies around the game board.

Next, the game is ready to have the agent enter the attack phase of their turn as a player. The Risk Game object asks the agent what territory it chooses to attack until it no longer gives any reply, indicating that it is finishing attacking for this turn. Whenever the agent responds with a territory that it would like to attack, the method tells the agent to proceed for as long as it wishes.

Once attacking is no longer favorable for the agent, it continues into the movement phase of the player's turn. The Risk Game's method asks the agent what armies it would like to move, as well as the sending and receiving territories. If the agent provides an answer, then the method tells the agent to proceed and request a Map object to transfer the armies.

3.7.3 Game Over

Whenever a player finishes their turn, the Risk Game object is ready to check for any game-ending conditions. First, it determines if any players have lost by requesting a collection of territories controlled by each player. If the size of a player's collection is zero, then that player has no claims to any territory on the game board and has therefore lost the game. The method moves any agents that fit this criterion from the list of active players to a list of losing players. When a player is removed from the list of active players, their turns are simply skipped. After the method has concluded checking for any losing players, it is much easier to check for any winning players. At this point, the only way a player could win is if they were the only player in the list of active players. If this is the case, then the player is removed from the list of active players and is placed into a list of winning players.

3.7.3.1 Tiebreakers

As mentioned in the rule changes section, games of RISK can often run a bit long, and a stochastic beam search lends itself to brevity. If after 100 turns each, no player has been placed into the list of winners, the Risk Game object is ready to stop the game, check for tie-breaking conditions, and report the winner or winners.

To start processing tiebreakers, the method begins with the list of active players, because all are potential winners at this point. The method counts the number of territories claimed by each player by using the Map object. The player with the most territories is moved into the list of winners, and the rest of the active players are moved into the list of losers. If there is still a tie because two or more players control the same number of territories, then the method requests the

sum of armies on all territories claimed by a player using a Map object. If there is no longer a tie, the agents are moved into the appropriate list of players.

Lastly, if there is still a tie, the method requests the number of armies each remaining player will receive at the start of their next turn. The player receiving the most armies at the start of their next turn shall be the winner, and any other players will be placed into the list of losers. If this still did not settle the tie, then all players with the same number of new armies coming in at the start of their next turn are placed into the list of winners, and all other remaining players are placed into the list of losers. The lists of both winners and losers are returned as the results of having played a game of RISK.

3.8 Agent Population

The Population class is the heart of the stochastic beam searching algorithm. This class is responsible for managing all of the different Agent objects that will slowly evolve and learn to play RISK with the help of the Tournament class. If the Population class is the heart of stochastic beam search, then the Tournament class would be the lungs providing the body with precious oxygen. The Population class can initiate a new collection of agents, add agents to the collection, clear the collection, create a new composite agent based on average agent definition, and divide the collection into matched sets that can compete against one another. Adding and removing agents from an internal collection is trivial and will not be discussed here.

3.8.1 Initiating a Population

The Population class needs to know the population size, a number of initial mutations, and a mutation multiplier. First, the class creates a new agent using default values. Next, it performs a given number of major mutations on the new agent. Additionally, it provides the

agent with a mutation multiplier to pass to each characteristic when mutating. The mutation multiplier is multiplied by a characteristic's adjustment value before adjusting the characteristic's value. By default, the number of initial mutations is ten, and the mutation multiplier is two. These high values ensure that each new agent is sufficiently random and unique from one another. This process repeats for each new agent until the desired population size is reached.

3.8.2 Creating a New Generation of Agents

Completing the next generation of agents requires two preconditions. The first is that some agents are already a part of the internal collection of agents, and the other is that the number of elements inside that collection is less than the desired population size. To create a new agent, the Population class starts by copying a random agent from the internal collection, provided that the agent existed in the collection when the method generating the next generation of agent was called. Next, the agent is requested to mutate, once again passing a mutation multiplier. By default, this mutation multiplier is one, but this will be adjusted by a Tournament object later. Once the agent has mutated, the new agent is added to the internal collection. The process repeats until the number of agents inside of the internal collection is equal to the desired population size.

3.8.3 Average Agent

For the purpose of tracking results, it is useful to look at the average values of each characteristic in a generation of agents. To accomplish this goal, a Population object creates a new agent. Next, it iterates through each category and characteristic in the agent's definition to set each value to 0. The next step involves iterating through each agent in the internal collection

and adding up the appropriate values and placing the sums in the appropriate characteristic of the new agent. To prevent the need to iterate through each characteristic a second time to divide by the total number of agents in the internal collection, this implementation distributes the division to each value before their addition. Refer to Equation 7 for further explanation.

$$c \times \sum f(x) = \sum c \times f(x)$$

Equation 7. Summation rule showing how a constant multiple of a function can be placed either inside or outside of the summation.

3.8.4 Dividing Agents into Matchups

To play in a tournament, agents must be divided into a discrete number of matchups to form the basis for the first round of tournament play. A Population object accomplishes this by creating a collection of collections of agents. The outside collection represents each matchup, and each inside collection contains the agents competing in that matchup. To create each matchup, the Population object iterates through all agents inside the internal collection, placing each agent into a matchup. Every few agents, according to a given group's size, the matchup is placed into the collection of matchups, and the next group of agents is placed into a new matchup. If there are not enough agents to complete the final matchup, then the Population object places as many agents as it can into the matchup. By default, the number of agents in each matchup is four.

3.9 Tournament Play

The Tournament class is the part of the program that takes the agents from the Population class and forces them to play RISK against one another to drive the optimization of the stochastic beam searching algorithm. A Tournament object manages a double-elimination tournament for a

population of agents. The tournament is double elimination to give each agent a second chance to win if an agent loses a game due to bad luck instead of a poor agent definition. The tournament does not seek to determine an overall winner but will finish once only 25% of the original population size remains.

3.9.1 Organizing a Tournament

Given a population of agents, a Tournament object can organize a tournament. First, it takes note of the original population size. Next, it creates three new collections. The first collection is for the winners of the most recent tournament round. The second collection is for the losers of the most recent tournament round. The last collection is for losers of all previous tournament rounds, hereafter named the loser's master collection.

Now that each collection is ready to be used, the tournament can really begin. First, a collection of matchups is provided by the agent population. For each matchup in the collection, the tournament creates a new Risk Game object and proceeds to play a game of RISK using the agents in the matchup. Once the game is finished, the winners of the game are added to the winner's collection, and the losers to the loser's collection.

Once all matchups have been processed and there are no more games to play, the Tournament object can prepare for the next round of the tournament. First, it clears the population of all agents stored in its internal collection. Next, it adds all agents currently in the winner's collection back into the population. Afterwards, it adds all agents currently in the loser's collection if and only if they are not also present in the loser's master collection. With each iteration, the number of agents in the loser's master collection will increase, and the number of agents in the population will decrease.

Once all eligible agents have been added back to the population, it is time to prepare for the next iteration, i.e., the next tournament round. First, all agents in the loser's collection are moved into the loser's master collection. Next, the loser's collection and the winner's collection are both cleared. This iterative process will continue until the population reaches 25% or less of its previous size.

3.9.2 Tournament Series

When implementing a stochastic beam search, organizing a tournament to find the top-performing agents in a population is only half of the story. The next steps are to create a new generation of agents based on the top performers of the previous generation and repeat the process until the agent definitions converge on a set of values that lead to winning gameplay. As the top performers continue to thrive and expand into the rest of the population, they will ideally converge onto novel and effective gameplay strategies for RISK.

To help promote a diverse initial population, a mutation multiplier of 1.75 is used when creating the first new generation of agents. However, keeping a high mutation multiplier for every generation could cause the agents to skip over characteristic values that lead to more optimal play. Therefore, the last generation of agents uses a mutation multiplier of 0.5. The intermediary generations use a mutation multiplier linearly interpolated from 1.75 to 0.5 across the 1500 generations. This implementation allows for high degrees of experimentation and diversity in early populations in conjunction with homogenous and precise fine-tuning in later populations.

3.10 Runtime Optimizations

Unfortunately, using stochastic beam search with a utility function that can take a few minutes to compute is very inefficient for runtime performance. Therefore, some optimizations must be made to help the algorithm converge on a successful agent definition within a reasonable time.

3.10.1 Parallel Processes

The first step towards a reasonable runtime is to recognize that each game of RISK runs independently of one another. Therefore, there is no reason to run each game on a single thread or process. Multi-processing proved to be a more substantial improvement than multi-threading in small-scale trial and error tests. Therefore, multi-processing was used in the final version of the application. This was easily achieved using python's *concurrent.futures* module that provided a high-level interface for asynchronously running each game of RISK, then compiling the results [14].

3.10.2 Data Caching

The second step towards a reasonable runtime was to cache data whenever possible. This list of examples is not exhaustive but is illustrative of the many areas that can benefit from caching. The first example is to store the rough estimate of any requested attack estimate in which the number of attackers or defenders is more than 30, thus preventing the need to calculate that estimate twice should the need arise.

Next, it is a common occurrence that multiple territories will share a connection with another territory, which can cause some operations affecting an action's impact score to be

repeated—namely, determining if a given territory is adjacent to an ally or enemy territory or if a territory borders another continent. Notice that when considering an action, the game board does not change, and therefore this data can be cached until the agent has finished evaluating the action.

3.10.3 Limited Dice Rolls

One final example of optimization is how the battle system returns a collection of dice rolls. The attack system will return a collection of random dice rolls of any size. Given that RISK rules dictate that the most dice used at any one time is when an attacker rolls three dice, it is obvious that this function would only ever return a collection of at most three elements. Therefore, it is quicker to skip the loop which produces the requested number of dice rolls and instead always generates a collection of three random dice rolls. This collection can then be truncated to the requested number of dice rolls. This method of generation proved slightly more performant over a large set of battles.

3.11 Error Recovery

Unfortunately, while producing results, a bug was discovered in a dependency of Python's *concurrent.futures* module [15]. The issue stemmed from a collection of processes increasing in size while iterating over the same collection. The issue caused an unrecoverable error in the program, forcing the program to crash and halt. The issue occurred semi-frequently, which prevented executions from completing all 1500 generations.

Successful execution of the program could take a few days to a week to complete, so time loss from failed executions was substantial. To recoup the time investment, the process was restarted using the partial results of the last execution as a starting point. The program was

provided the file path of the last output describing the characteristics of the average agent of the last generation of agents before failure. The program then creates a new agent based on the file's data and then uses that agent as the basis for the rest of the population.

The diversity of agents within the new population is minimal in this circumstance because all agents are based on the same ancestral agent. The new population is created with a mutation multiplier of five to help promote diversity in agents in these situations.

Chapter 4. Results

The agents converged on a workable set of characteristics that exploited changes in the game rules, specifically as they relate to tying. Agents learned that forcing a tie and winning via the tie-breaking rules is a safer and more consistent way of winning RISK as implemented. Because the system has no way to punish this behavior, the agents changed the game's goal from "winning" to "not losing."

General trends suggest the agents learned to prefer defensive behavior, likely not to jeopardize their chance of winning via tie-breaking. Additionally, agents learned not to deviate from this strategy out of fear of retaliation from the other agents. The agents learned to prefer attacking more aggressive and threatening agents. Furthermore, to help ensure a tie between the agents, agents would actively try to spoil the aggressive strategies of other players preventing any one player from gaining an upper hand on the gameboard.

The program crashed five times during the final run at generations: 265, 523, 566, 714, and 841. The errors introduced a considerable amount of noise into the data making it tricky to determine trends in the data. However, many of the variances caused by the sudden change in diversity were quickly course-corrected in line with previous trends. Additionally, the sudden change in behavior caused by the higher-than-normal mutation multiplier would occasionally knock the agents out of a local optimum and promote the development of new strategies.

4.1 Placement Category

The agents learned to favor continental thinking when placing armies onto the game board. The agents showed a slight preference for placing armies along the border of a continent (Table 4, Figure 8). There was a consistently strong preference to not place armies in areas

surrounded by ally-controlled territories and instead place them on territories with many enemy connections (Figure 6, Figure 10). This behavior is expected and reflects either an offensive strategy or a “candy security” approach to defense [16]. Next, the “Placement Bias Multiplier,” which was a way to introduce diminishing returns on placing armies on a territory, ended fairly low at only 83.4% (Table 4). A closer inspection reveals that the value was initially very high, near 100%, but the error recovery events incidentally pushed the value significantly lower than previous levels (Figure 11). The agents failed to bring the value back up to near 100%, but the value was trending upwards for approximately 500 generations (Figure 11).

4.2 Place Continent Category

The agents learned to strongly prefer placing armies in the North American, South American, and European continents (Table 5). The Asian continent also held a slightly positive value but had no strong trend upwards or downwards (Figure 13). By contrast, the agents had slight disfavor for African and Australian continents and continued to trend downward across the final generations (Table 5, Figure 12, Figure 14).

I suspect that the agents settled on these preferences because the preferred continents are highly interconnected to the other continents (Figure 2), allowing players in control of those continents to claim territory on adjacent continents quickly. While claiming these individual continents did little to help the attacking player, they significantly hurt the defending player by removing any continental bonus they may have received on their next turn. Similarly, the agents learned to avoid continents that were more isolated from the other continents (Figure 2, Table 5).

The only outlier to this theory is the Asian continent (Figure 13). However, considering that the continent is so interconnected with the others, it is unlikely to have a single-player claim

control of all territories on the continent. Therefore, if the agents' primary goal of placing armies in the preferred continents is to orchestrate attacks that spoil continent bonuses for other players, it is unlikely to be necessary on the Asian continent.

4.3 Attack Category

The agents learned that attacking is rarely worth the lost armies and avoided hostile interactions whenever possible. Every characteristic value of attacking was negative (Table 6), except when combined with the characteristic values from the Attack Continent category (Table 7). The agent learned to strongly dislike conquering continents (Table 7), contrary to what may be expected. However, considering that the agent was likely aimed at just surviving until the tie-breaking rules took effect, not conquering a continent prevented the player from looking like a threat to the others. Not conquering continents and allowing the other agent's preference for attacking larger players kept themselves under the radar of the other players (Table 9) and, therefore, more likely to win.

Attacks tended to be all or nothing assaults, with attacks not called off until only 8.2% of the original number of attacking armies remained (Table 6). However, this rarely mattered because the estimated chance of success needed to be 78.2% or higher before the agent proceeded with the attack (Table 6, Figure 25). This low tolerance for risk allowed the minimum impact score to hover around relatively low levels. Therefore, I conclude that agents did not need to gain much from a successful attack for it to consider worth engaging, as long as the chance of success was high.

4.4 Dice Roll Analysis

Agents found that being unpredictable was more valuable than any particular strategy for dice rolls. The number of attacking dice bounced around 2, wildly swinging ± 0.5 to round to either 1 or 3 and settling on 2 in the meantime (Table 6, Figure 20). Similarly, the number of defending dice bounding around 1.5, rounding to either 1 or 2 (Table 6, Figure 23Figure 20). The values determining how many dice to roll when attacking and defending failed to reach an equilibrium because no strategy dominated any other. The randomness provided by the dice rolls likely made these values difficult to pin down.

Following Iain Hart's methodology [11], I was able to compare the results of 5,000 different attacks using a maximum and minimum attackers strategy (Figure 50, Figure 51). These tests assume the defending territory's player uses a maximum defenders strategy. As expected from Nick Berry's results [10], the maximum attackers strategy typically outperforms the minimum attackers strategy (Figure 52). Surprisingly, the minimum attackers strategy appears to perform at least as well whenever the number of attacking armies greatly outnumbered the number of defending armies (Figure 52).

The "Percent Change Between Attacker Strategies" graph indicates that as the number of attackers increases, the results differ minimally as the number of attackers increases (Figure 52). The large section of red in the top right corner indicates a 0% change because the values on both graphs are also zero. This graph shows how the minimal attackers strategy can be viable when paired with a highly defensive strategy that builds up armies in mass before attacking.

I used the same analysis process to examine a maximum and minimum defenders strategy (Figure 53, Figure 54). For this experiment, the attacker uses a maximum attackers strategy. The

“Percent Change Between Defender Strategies” graph indicates that the strategy reduces the success chance for both attackers and defenders, making the outcome more random (Figure 55). This effect can be advantageous to the defender if the attacker has only a slight advantage. The large section of green in the bottom left corner indicates a 0% change because the values on both graphs are also zero.

4.5 Attack Continent Category

As the agents preferred to play defensively, all characteristic values settled on negative values (Table 7). Therefore, the discussion around these characteristics is more about what the agents disliked the least rather than their actual preferences. To that end, the European continent had a strong trend downwards and was the most disliked continent (Figure 33). When combined with the knowledge from the Placement category’s characteristics (Table 4), it appears that a core strategy of the agents was to claim Europe during the game setup (Figure 15), place many armies there to hold it, use it as an attack vector for spoiling the continental bonuses of other players (Table 6), and to avoid the continent once another player had claimed it (Figure 33, Table 6).

Australia and South America were similarly disliked (Figure 32, Figure 35), likely because of their limited connections to other continents (Figure 2). Africa, Asia, and North America were the least bad options when considering attacks (Table 6), likely because allowing another player to capture any of these continents would likely jeopardize the game ending in a tie and foiling agents aiming to win by tie-breaking rules. The North American value ended up slightly higher than the others (Table 6), but an arbitrary rise following an error recovery event

can explain the discrepancy (Figure 34). The value quickly dropped down to match previous trends but failed to stabilize before reaching the 1500th generation.

4.6 Movement Category

As expected, the agents learned it is preferable to move armies away from territories surrounded by many territories controlled by the same player (Figure 36, Figure 41). They instead move armies onto territories adjacent to many enemy-controlled territories or along continental borders (Figure 39, Figure 41). Furthermore, they strongly preferred moving armies onto territories with many armies (Figure 38). This preference suggests that the agents preferred to stockpile armies onto a single territory for an attack rather than attack a single defending territory using armies from two attacking territories in a combinatorial attack.

The safe transfer rate was consistently high, around 90%, but started to drop sometime after the 1200th generation and finalized at only 67% (Table 8, Figure 44). By contrast, the risky transfer rate was much more irregular but saw a steady upward trend following the 1000th generation concluding at almost 89% (Table 8, Figure 43). These changes coincide with changes in the “Safe” and “Risky” characteristic values from the “Preference” category (Figure 33, Figure 34).

Interestingly, the agents settled on a negative value for the “Connection Bias” characteristic (Table 8), suggesting that agents preferred isolationist behavior. This conclusion is contrary to the theory that a core strategy of the agents was to spoil enemy plans of conquest. However, closer inspection reveals that an error recovery event arbitrarily lowered the value (Figure 40). The value showed little to no trend following any error recovery event. The lack of

trends or significant changes suggests that the consideration was unimportant for developing the strategy.

4.7 Preference Category

The preference category is more difficult to decipher than the other categories because they influence a wide variety of behaviors. Additionally, the error recovery events introduced much noise to the data, making it challenging to identify specific trends (Figure 46, Figure 49, Figure 47, Figure 48). Broadly speaking, the agents tended to favor the Risky characteristic over the Safe characteristic (Table 9). This favoritism stands in contrast to the behavior implied by characteristics in the other categories. The inconsistent framing of the behavior may be explained by the same values determining what behaviors are riskier or safer than others, which may have influenced the results.

The agents showed a continuous preference for engaging with larger, i.e., more threatening, players (Figure 46, Figure 49). The “Larger” characteristic showed a small but steady upward trend across all generations (Figure 46). By contrast, the “Smaller” characteristic shows no consistent trend at any point, often radically changing after each error recovery event (Figure 49). The value finally settled at just under -1, indicating a slight dislike for engaging with smaller, i.e., less threatening, players (Table 9). However, the value appears to be converging around zero, but even this is uncertain because of the high degree of noise in the data (Figure 49). An alternate theory is that the agents recognized the “Smaller” characteristic as redundant to the “Larger” characteristic, thereby treating it as unimportant by comparison.

Chapter 5. Conclusions & Future Research

The agents in this project did not develop a novel strategy that would allow them to compete against human players at a competitive level. However, the agents were able to provide a solid and easily extendable approach to playing turn-based strategy games. Simultaneously, the agents were able to disprove some commonly held beliefs regarding RISK strategy. Likewise, the agents' characteristics reinforced other widely held expectations and strategies.

The agent definition model proved an effective and easily extendable way of integrating complex learning behavior. Stochastic Beam Search also proved an effective way to coordinate behaviors and adjust characteristic values. It is unclear if the double-elimination tournament was the most effective method of determining the relative fitness of each agent as there was no comparison. However, the double-elimination tournament did not appear to hinder the Stochastic Beam Search because the agent's characteristics converged onto consistent values and trends.

5.1 Comparison to Blogger Strategies

From a high-level perspective, the agents learned to play hyper-defensively. They preferred to reactively spoil other players' strategies to keep them from gaining a significant advantage and exploit the tie-breaking rules to force an easy victory. The defensive posturing and continental thinking mirror the conclusions found by the bloggers (Table 4, Table 5, Table 6, Table 7). Settling the dispute between the blogger strategies, these agents learned that maintaining borders is best achieved by accumulating armies along the continental border rather than pre-emptive strikes into other players' territories (Table 4, Table 8).

Contrary to the popular beliefs espoused by the bloggers, the Australian and South American continents were not desirable (Table 5, Table 7). This conclusion is likely because the

agents viewed these continents as too isolated to outmaneuver the other players effectively (Figure 2). Additionally, the army bonus from their control may not be worth the additional effort to maintain control over the required territories. Furthermore, a detailed analysis of dice roll strategies reveals that a minimal attacker strategy can be a more effective strategy if the attacker already has a significant advantage (Figure 52). Likewise, a minimal defender strategy can be helpful when the number of attackers and defenders is nearly equal, but the attacker holds a slight advantage (Figure 55).

5.2 Future Research

Future research could be dedicated to increased optimization for Monte Carlo Search, allowing forward-thinking agents to navigate a game as complex and branching as RISK. Future research could also adapt the current system to include more considerations, including a full turn heuristic that permits more coordination between decisions in separate turn phases. Lastly, the agent characteristic model and stochastic beam search could be applied to other strategy games—specifically, strategy games whose branch factors make forward-thinking artificially intelligent agents prohibitively slow for effective use.

This thesis has highlighted the need to apply more research to the game of RISK. Lessons learned here may be applied to other strategy games and problems whose complexity and high branching factor limit more conventional methods' effectiveness.

REFERENCES

- [1] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd ed., Pearson, 2009.
- [2] P. Brothers, "RISK: Instructions," Hasbro, 1993. [Online]. Available: <https://www.hasbro.com/common/instruct/risk.pdf>. [Accessed 14 February 2021].
- [3] K. Veronese, "The Origins and Evolution of the Strategy Board Game RISK," 30 March 2012. [Online]. Available: <https://gizmodo.com/the-origins-and-evolution-of-the-strategy-board-game-ri-5897532>. [Accessed 17 September 2021].
- [4] Ultra BoardGames, "Tips to win Risk," 2020. [Online]. Available: <https://www.ultraboardgames.com/risk/tips.php>.
- [5] M. Wolf, "An Intelligent Artificial Player for the Game of Risk," Darmstadt University of Technology, Darmstadt, 2005.
- [6] GameServer Team, "19 Tips, Tricks, and Strategies to Win at Risk (the Game)," Ropcaf, 16 July 2021. [Online]. Available: <https://www.gamesver.com/19-tips-tricks-and-strategies-to-win-at-risk-the-game/>. [Accessed February 2022].
- [7] J. Shoemaker, "9 Tips for Stepping up Your Risk Board Game Strategy," Bar Games 101, 14 July 2020. [Online]. Available: <https://bargames101.com/risk-strategy/>. [Accessed February 2022].
- [8] G. Robinson, "The Strategy of Risk," May 2009. [Online]. Available: <https://web.mit.edu/sp.268/www/projects.html>. [Accessed December 2021].
- [9] M. Parsley, "How To Win Risk Board Game: Rules, Strategy And Tips," Dice Glory, 2 December 2021. [Online]. Available: <https://diceglory.com/how-to-win-risk/>. [Accessed February 2022].
- [10] N. Berry, "RISK Analysis," November 2011. [Online]. Available: <https://datagenetics.com/blog/november22011/index.html>. [Accessed 28 February 2021].
- [11] I. Hart, "Risk Combat Statistics," [Online]. Available: <http://www.cs.man.ac.uk/~iain/riskstats.php>. [Accessed 14 February 2021].
- [12] E. Blomqvist, "Playing the Game of Risk with an AlphaZero Agent," Kth Royal Institute of Technology's School of Electrical Engineering and Computer Science, Stockholm, 2020.
- [13] M. Al-Ani, "Asexual versus Sexual Evolutionary Algorithms for Numerical Optimization," University of Baghdad, Baghdad, 2002.
- [14] Python Software Foundation, *concurrent.futures — Launching parallel tasks*, 3.10.2 ed., Wilmington, Delaware: Python Software Foundation, 2022.
- [15] Python Bug Tracker, "Issue43498," The Python Software Foundation, 15 3 2021. [Online]. Available: <https://bugs.python.org/issue43498>. [Accessed December 2021].
- [16] W. S. Kevin Mitnick, "The Art of Deception," in *The Art of Deception*, Indianapolis, Indiana: Wiley Publishing Inc., 2002, p. 79.

APPENDICES

Appendix A: Agent Definition

Agent Characteristic Descriptions					
Characteristic Name	Description				
	Final Value	Initial Value	Adjustment Amount	Lower Limit	Upper Limit

Table 3. Key explaining how to read an Agent Characteristic chart.

Placement Category

Ally Adjacent	Placing a unit on a territory connected to a territory controlled by the same player				
	-8.843	0	1	-10,000	10,000
Anywhere	Placing a unit anywhere				
	-3.09	0	1	-10,000	10,000
Border Adjacent	Placing a unit in a territory that borders a country in a different continent				
	2.688	0	1	-10,000	10,000
Connection Bias	Placing a unit on a territory with connections to multiple other countries, +value per connection				
	0.208	0	0.25	-10,000	10,000
Enemy Adjacent	Placing a unit on a territory connected to a territory controlled by a different player				
	-0.912	0	1	-10,000	10,000
Placement Bias Multiplier	Placing a unit where there already are other units, value^(armies on territory)				
	0.834	0.75	0.05	0	1

Table 4. Agent Characteristic chart for the Placement category.

Place Continent Category

Africa	Preference for placing an army in Africa				
	-21.979	0	10	-10,000	10,000
Asia	Preference for placing an army in Asia				
	27.188	0	10	-10,000	10,000
Australia	Preference for placing an army in Australia				
	-30.790	0	10	-10,000	10,000
Europe	Preference for placing an army in Europe				
	152.420	0	10	-10,000	10,000
North America	Preference for placing an army in North America				
	201.545	0	10	-10,000	10,000
South America	Preference for placing an army in South America				
	126.276	0	10	-10,000	10,000

Table 5. Agent Characteristic chart for the Place Continent category.

Attack Category

Ally Adjacent	Attacking a territory connected to another territory controlled by the attacking player				
	-0.997	0	1	-10,000	10,000
Anywhere	Attacking anywhere				
	-1.924	0	1	-10,000	10,000
Attack Dice Count	Preferred (max) number of dice to roll when attacking				
	1.570	2	0.5	1	3
Border Adjacent	Attacking a territory on the border of a different continent				
	-5.632	0	1	-10,000	10,000
Capture Continent	Attacking a territory that will give this player control over all territories on a continent if the attack is successful				
	-11.001	0	1	-10,000	10,000
Defend Dice Count	Preferred (max) number of dice to roll when defending				
	1.522	1.5	0.5	1	2
Minimal Remaining Percent	The amount of units lost before calling off an attack, expressed as a percentage of the amount of units at the start of the attack				
	0.082	0.1	0.05	0	1
Minimal Success Chance	The minimal amount of estimated chance of successful attack necessary for an attack to be considered viable				
	0.782	0.5	0.05	0	1
Minimum Impact Score	Minimum impact score that an attack must achieve before committing to the attack				
	-0.111	0	1	-10,000	10,000
Safe Threshold	The minimal amount of estimated chance of a successful attack to consider an attack safe; below this amount is considered risky				
	0.646	0.95	0.05	0	1
Survivorship Bias	The estimated value of surviving attackers, *value per unit				
	-0.020	0	0.1	-10,000	10,000
Targets Destroyed Bias	The estimated value of destroying defenders, *value per unit				
	-1.643	0	0.1	-10,000	10,000

Table 6. Agent Characteristic chart for the Attack category.

Attack Continent Category

Africa	Preference for attacking Africa				
	-44.508	0	10	-10,000	10,000
Asia	Preference for attacking Asia				
	-46.201	0	10	-10,000	10,000
Australia	Preference for attacking Australia				
	-62.832	0	10	-10,000	10,000
Europe	Preference for attacking Europe				
	-145.322	0	10	-10,000	10,000
North America	Preference for attacking North America				
	-34.483	0	10	-10,000	10,000
South America	Preference for attacking South America				
	-85.917	0	10	-10,000	10,000

Table 7. Agent Characteristic chart for the Attack Continent category.

Movement Category

Ally Adjacent	Moving a unit on a territory connected to a territory controlled by the same player				
	-15.790	0	1	-10,000	10,000
Anywhere	Moving a unit anywhere				
	-7.707	0	1	-10,000	10,000
Bigger Territory	Moving units onto a territory with more units				
	4.052	0	1	-10,000	10,000
Border Adjacent	Moving a unit in a territory that borders a country in a different continent				
	5.313	0	1	-10,000	10,000
Connection Bias	Moving a unit on a territory with connections to multiple other countries, +value per connection				
	-5.249	0	0.25	-10,000	10,000
Enemy Adjacent	Moving a unit on a territory connected to a territory controlled by a different player				
	8.033	0	1	-10,000	10,000
Minimum Impact Score	Minimum impact score that a movement must achieve before committing to the movement				
	-1.600	0	1	-10,000	10,000
Risky Transfer Rate	Percentage of units to transfer if the movement is considered risky				
	0.888	0.25	0.05	0	1
Safe Transfer Rate	Percentage of units to transfer if the movement is considered safe				
	0.670	0.75	0.05	0	1
Smaller Territory	Moving units onto a territory with fewer units				
	-6.015	0	1	-10,000	10,000

Table 8. Agent Characteristic chart for the Movement category.

Preference Category

Larger	Preference to attack larger players				
	3.924	1	0.25	-10,000	10,000
Risky	Preference for risky actions				
	1.559	1	0.25	-10,000	10,000
Safe	Preference for safe actions				
	-1.013	1	0.25	-10,000	10,000
Smaller	Preference to attack smaller players				
	0.475	1	0.25	-10,000	10,000

Table 9. Agent Characteristic chart for the Preference category.

Appendix B: Results Graphs

Placement Category

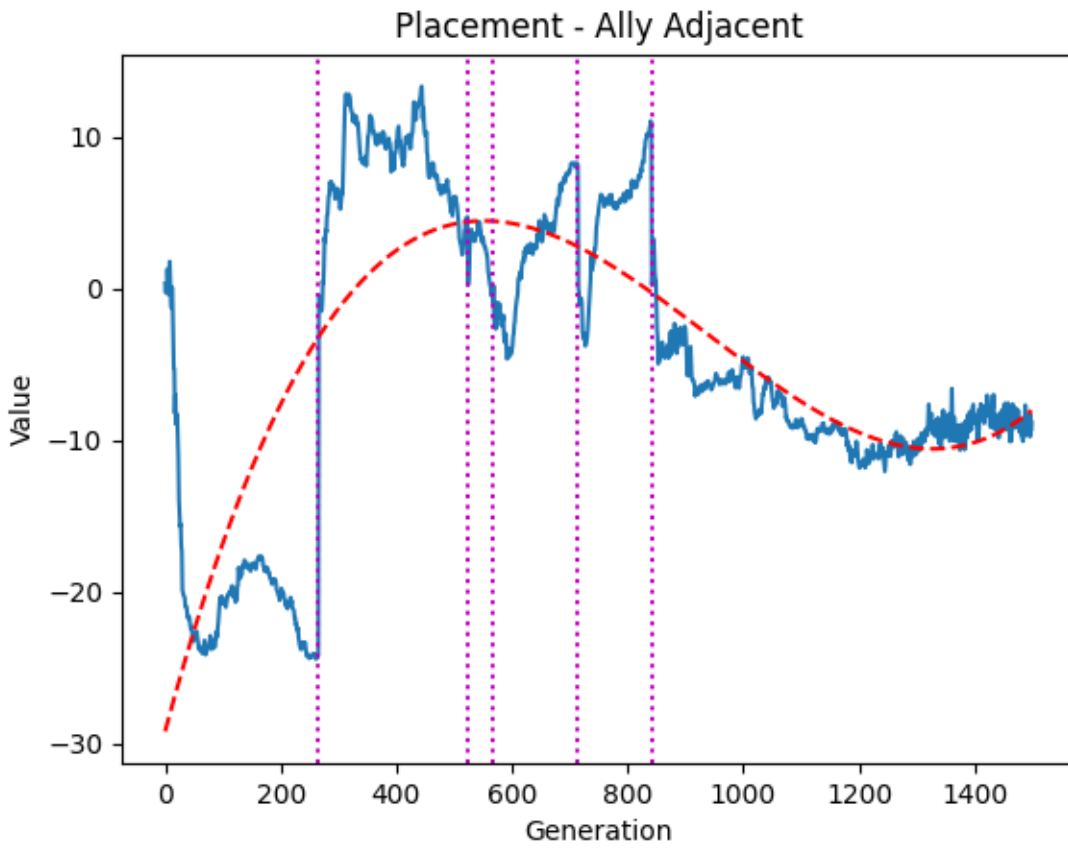


Figure 6. Graph of how the "Ally Adjacent" characteristic from the "Placement" category changed over time.

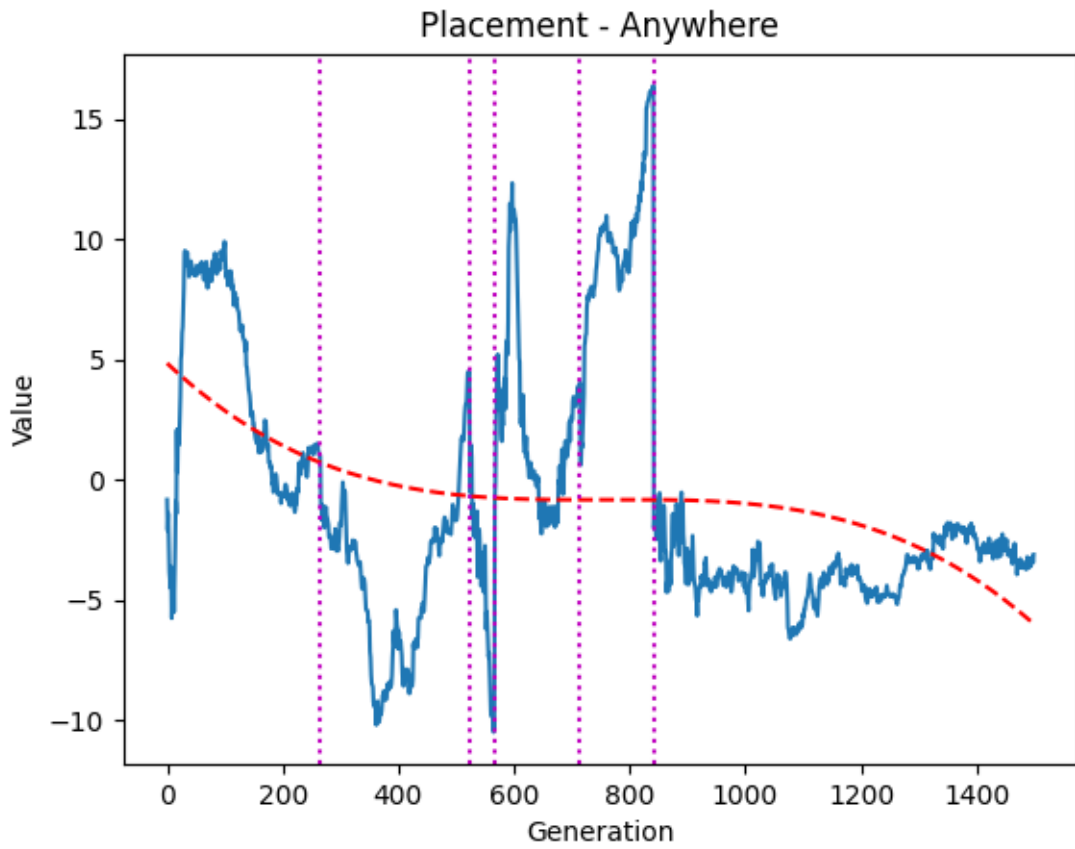


Figure 7. Graph of how the "Anywhere" characteristic from the "Placement" category changed over time.

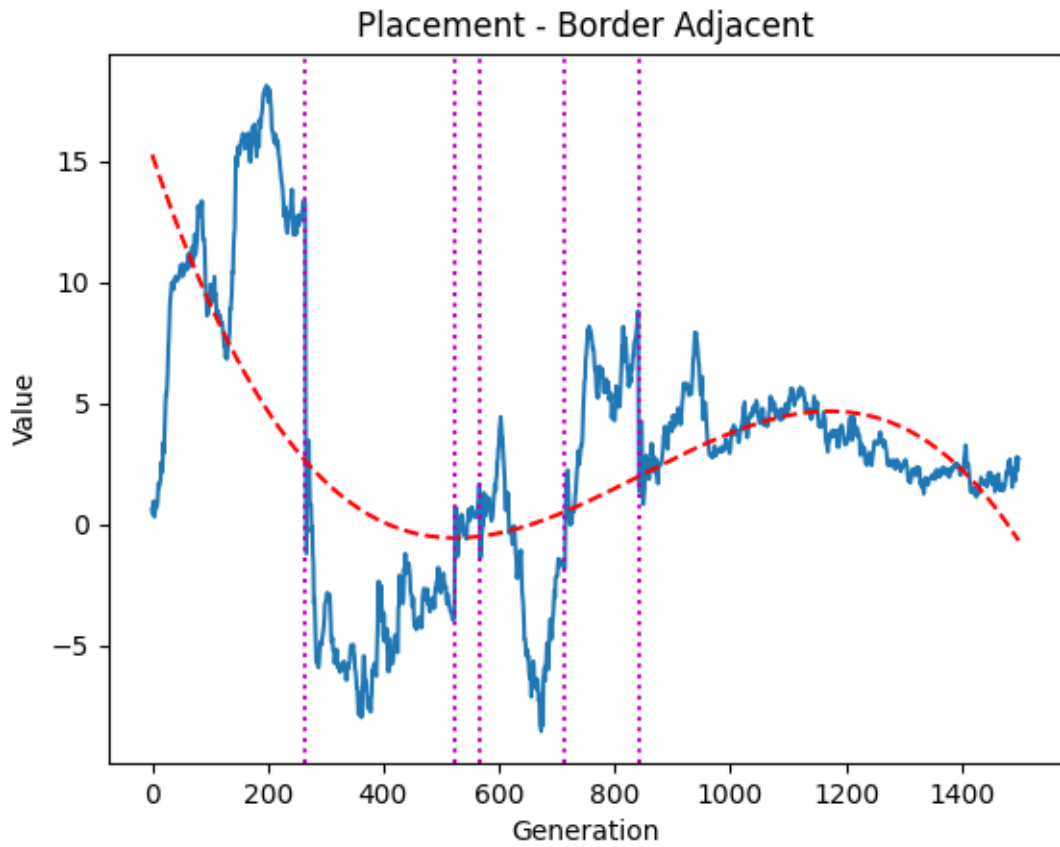


Figure 8. Graph of how the "Border Adjacent" characteristic from the "Placement" category changed over time.

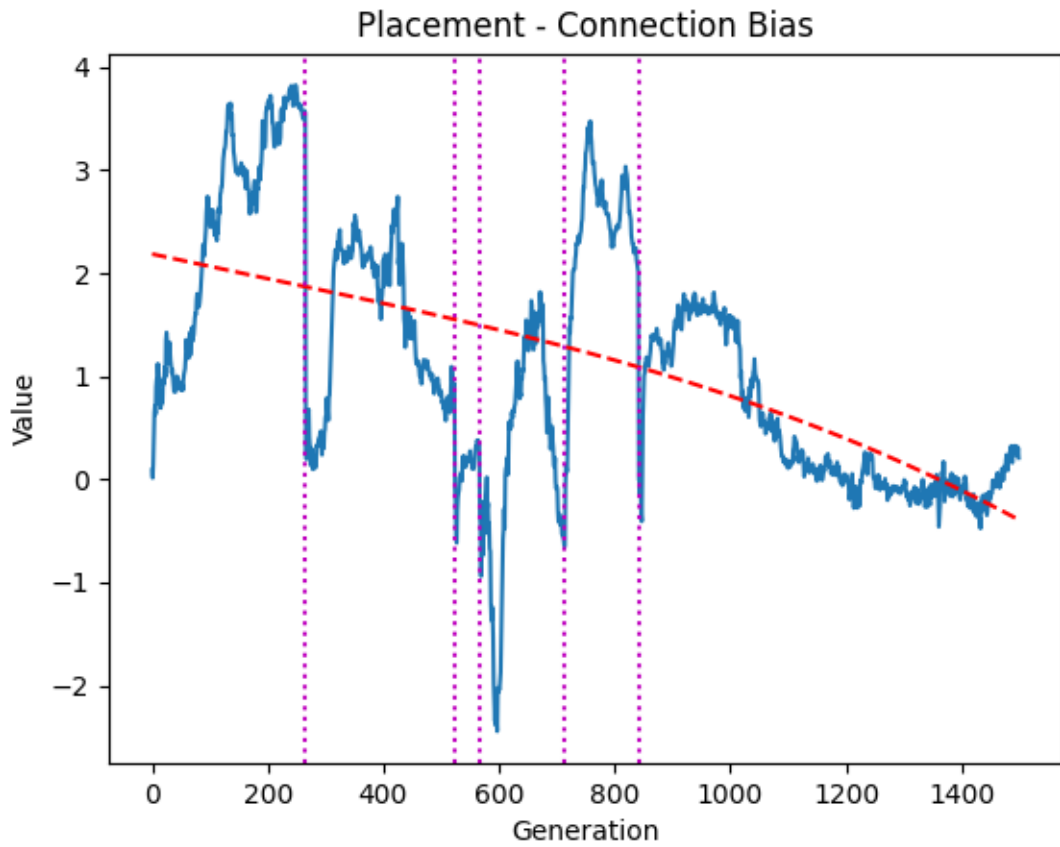


Figure 9. Graph of how the "Connection Bias" characteristic from the "Placement" category changed over time.

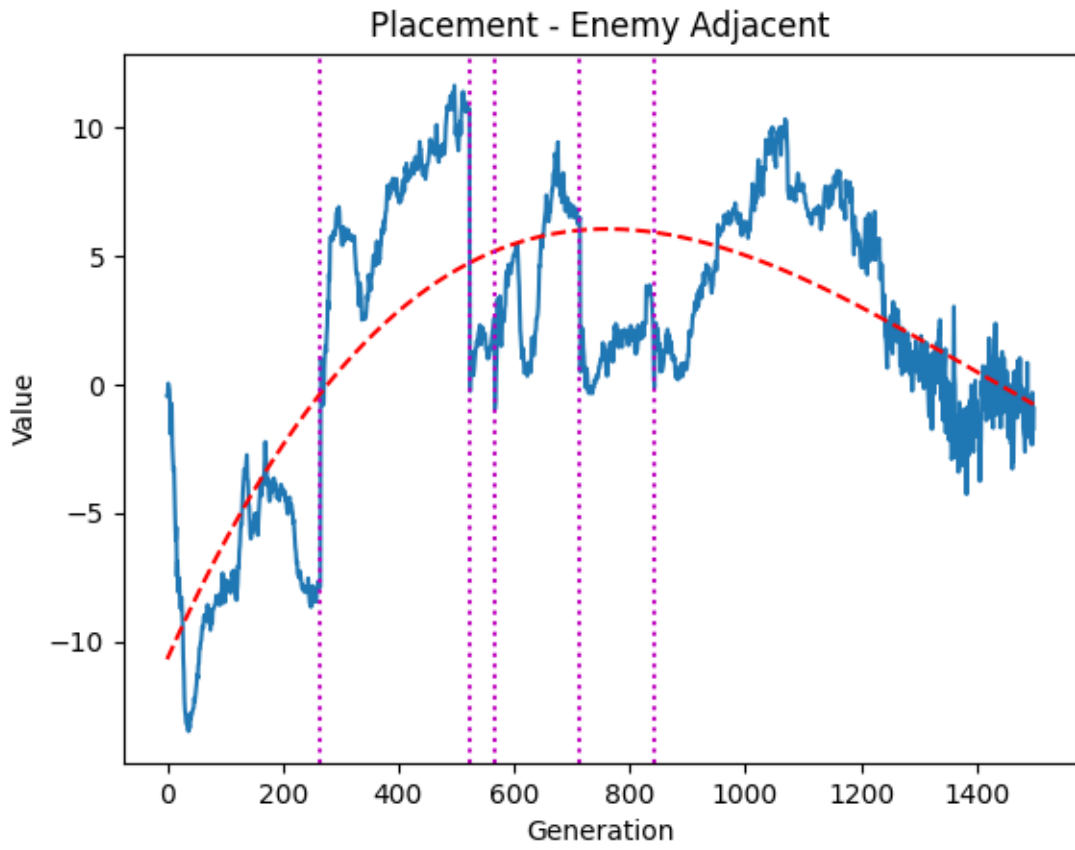


Figure 10. Graph of how the "Enemy Adjacent" characteristic from the "Placement" category changed over time.

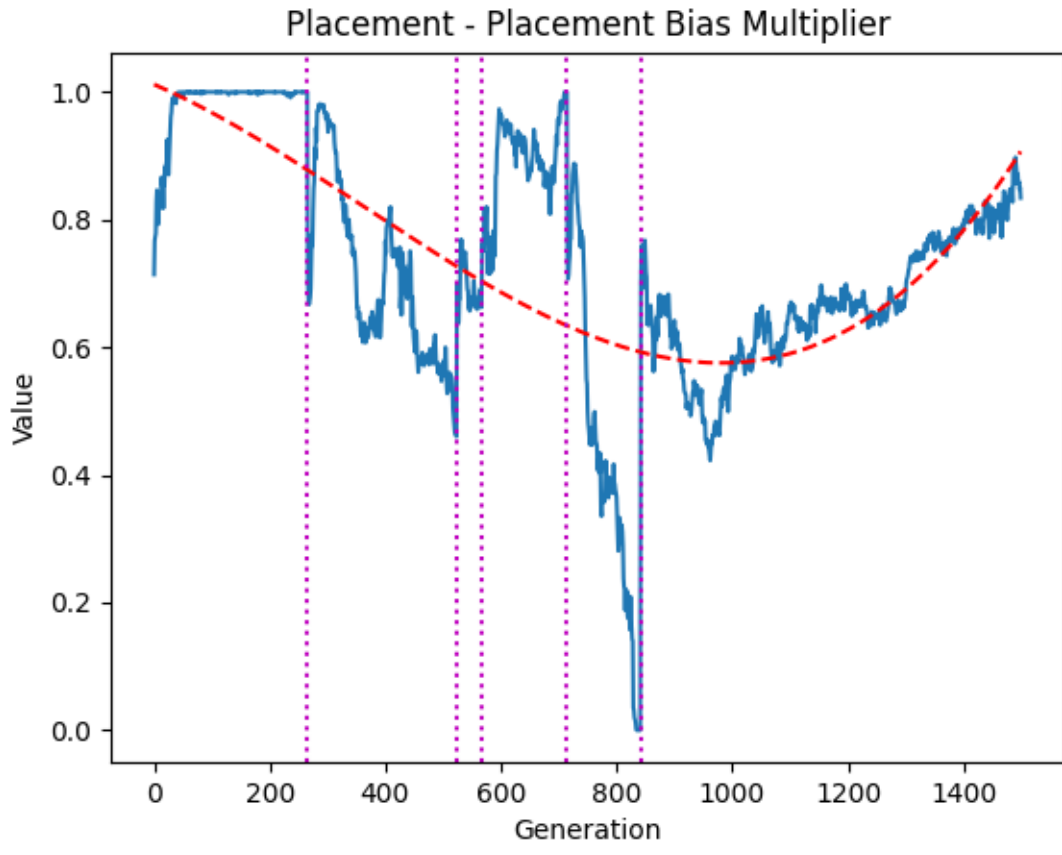


Figure 11. Graph of how the "Placement Bias Multiplier" characteristic from the "Placement" category changed over time.

Place Continent Category

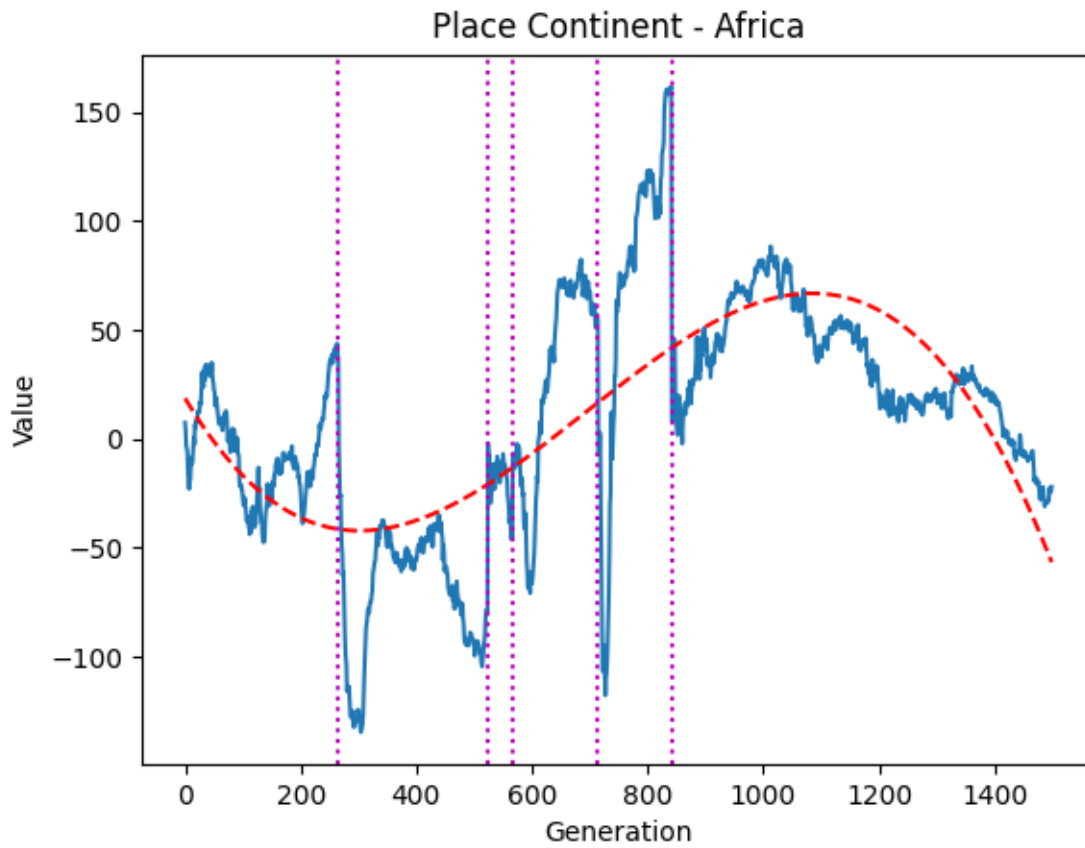


Figure 12. Graph of how the "Africa" characteristic from the "Place Continent" category changed over time.

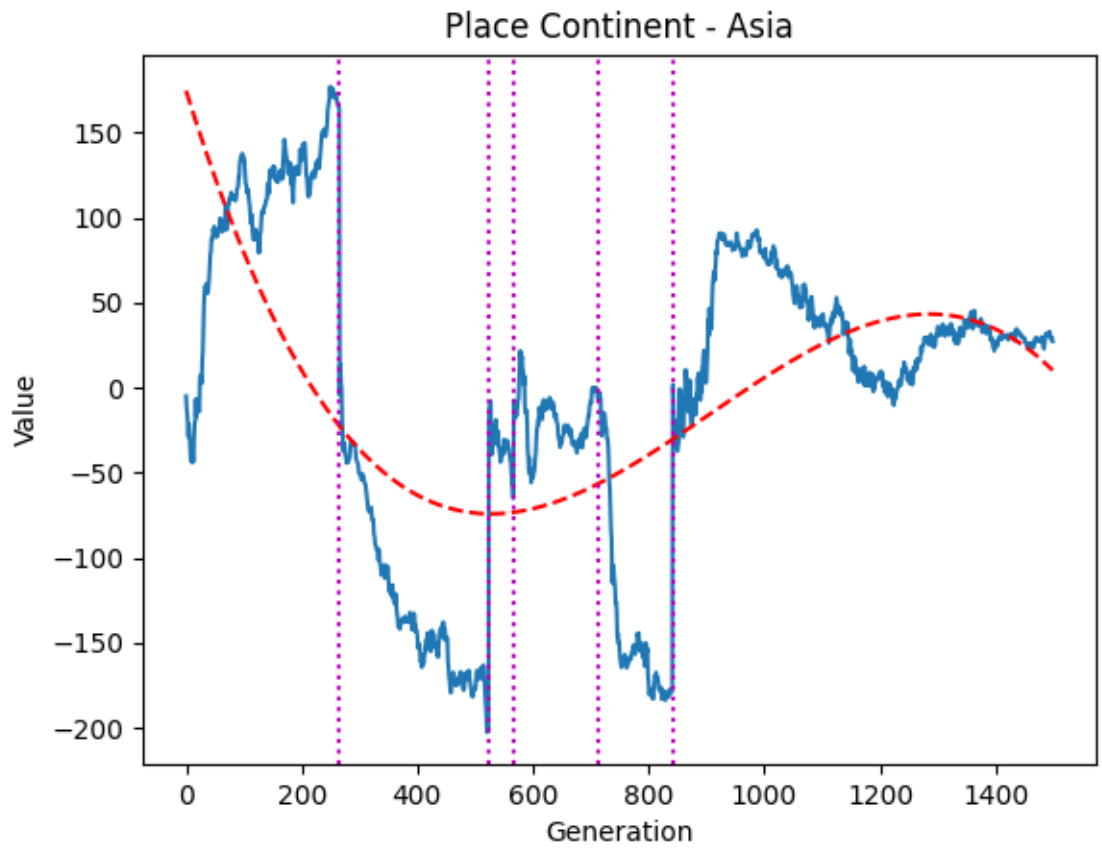


Figure 13. Graph of how the "Asia" characteristic from the "Place Continent" category changed over time.

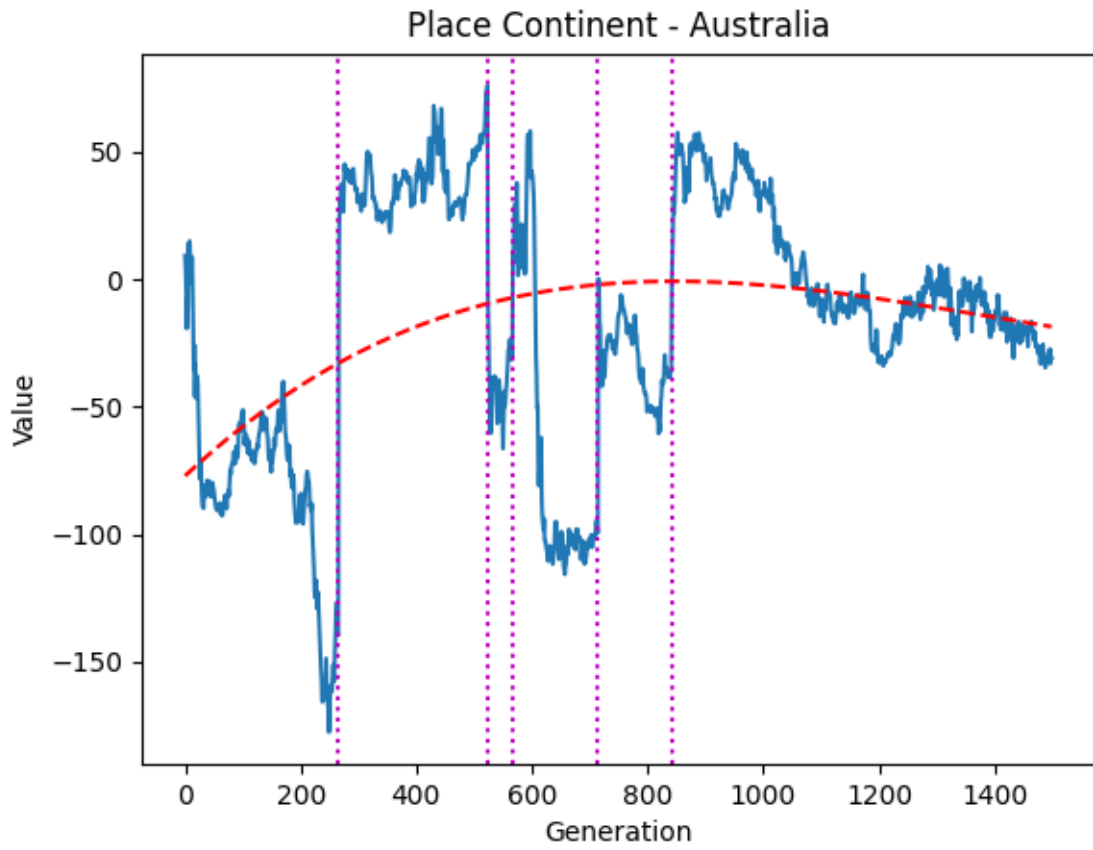


Figure 14. Graph of how the "Australia" characteristic from the "Place Continent" category changed over time.

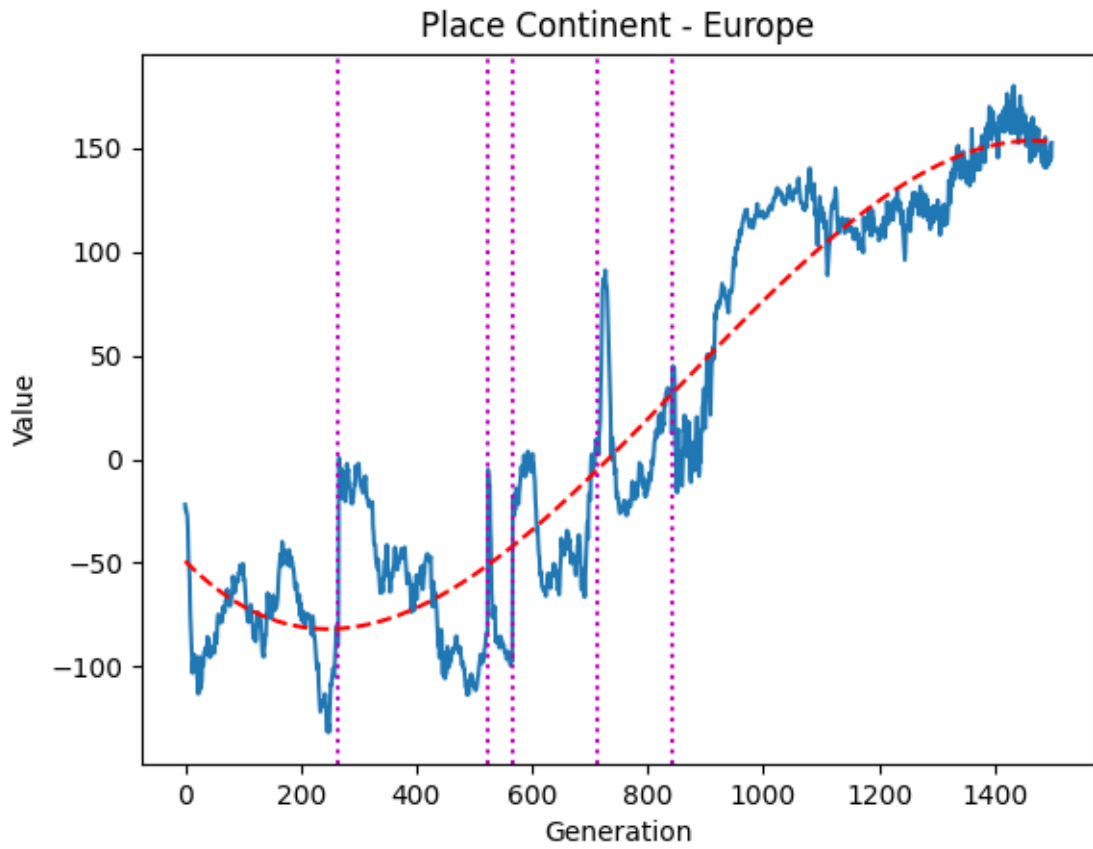


Figure 15. Graph of how the "Europe" characteristic from the "Place Continent" category changed over time.

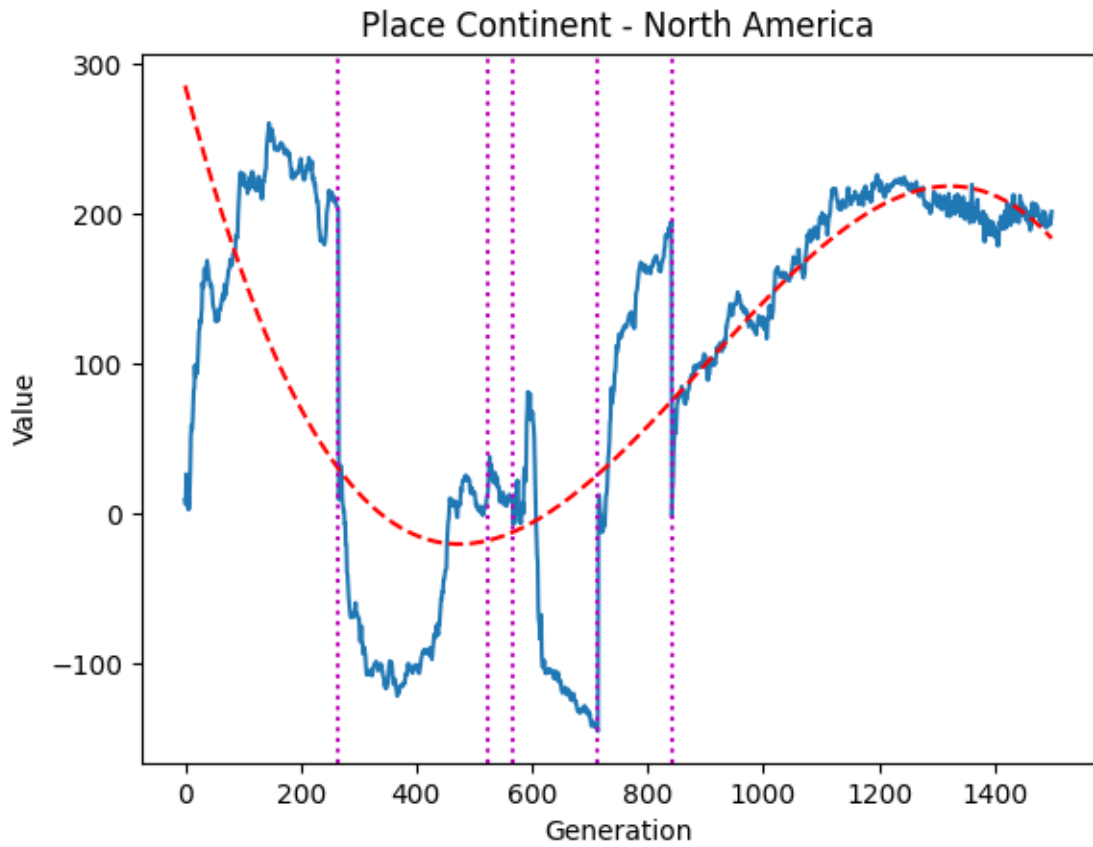


Figure 16. Graph of how the "North America" characteristic from the "Place Continent" category changed over time.

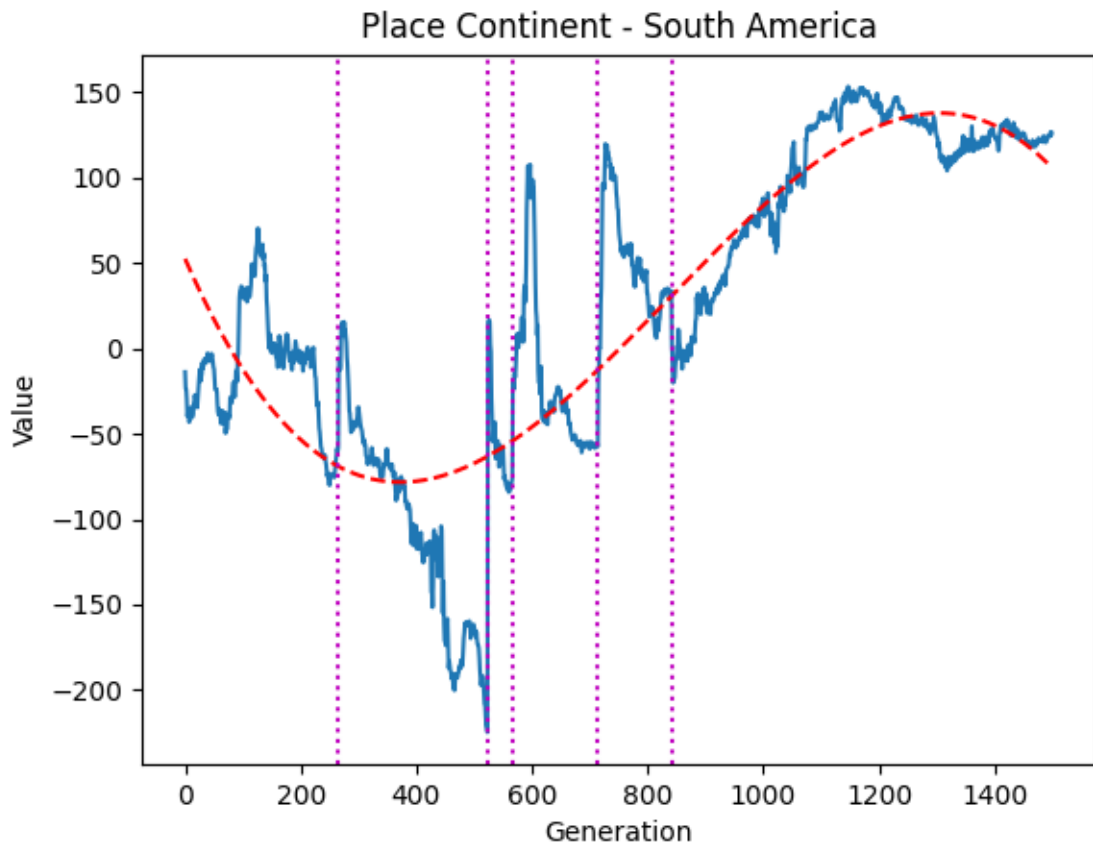


Figure 17. Graph of how the "South America" characteristic from the "Place Continent" category changed over time.

Attack Category

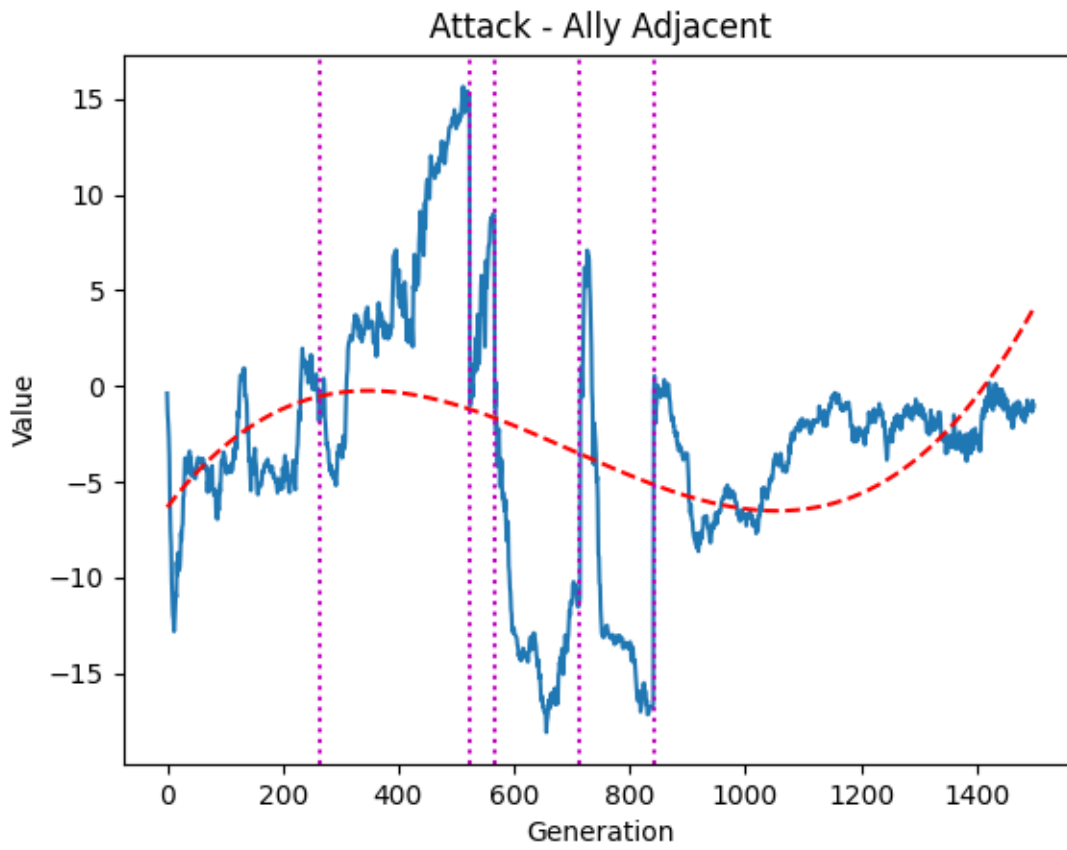


Figure 18. Graph of how the "Ally Adjacent" characteristic from the "Attack" category changed over time.

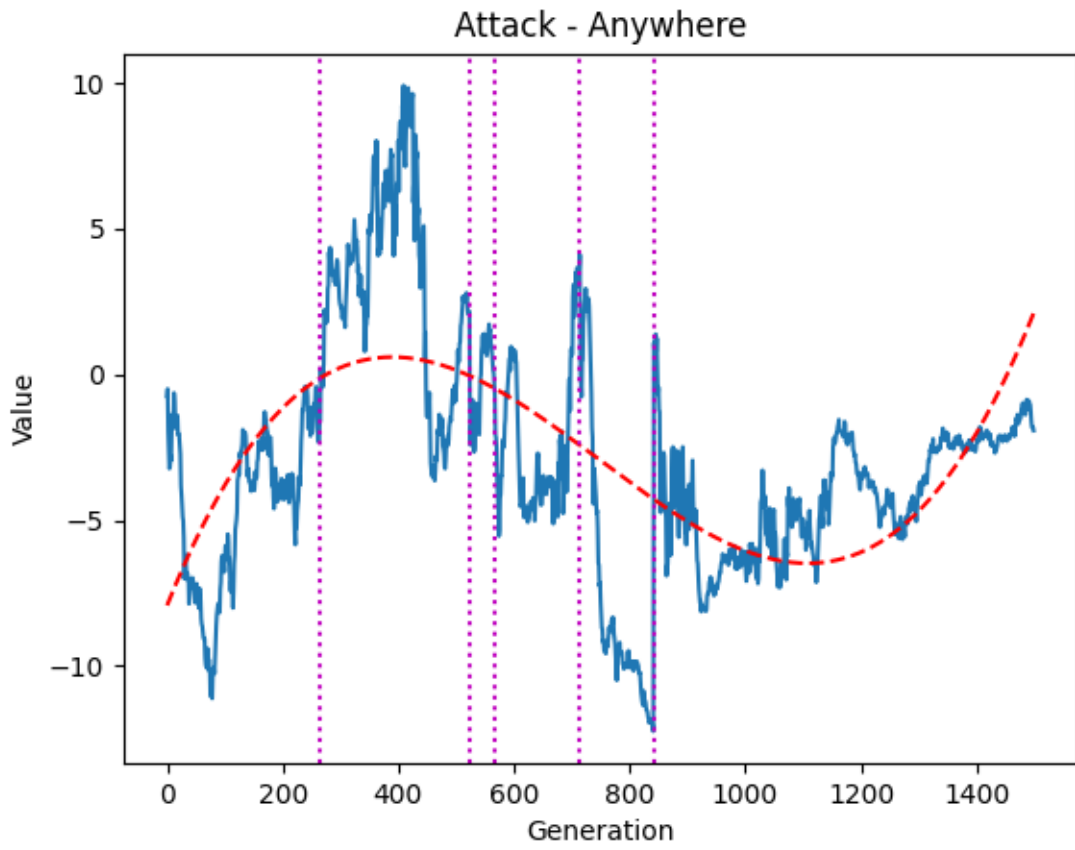


Figure 19. Graph of how the "Anywhere" characteristic from the "Attack" category changed over time.

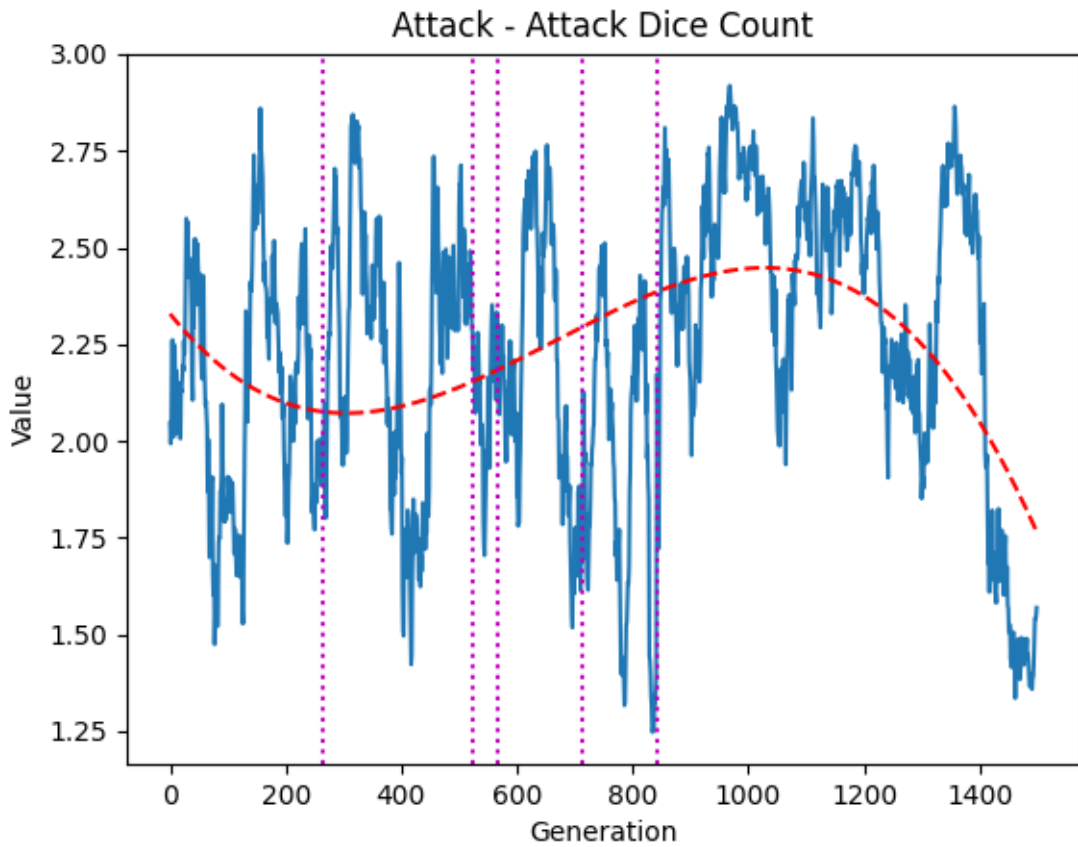


Figure 20. Graph of how the "Attack Dice Count" characteristic from the "Attack" category changed over time.

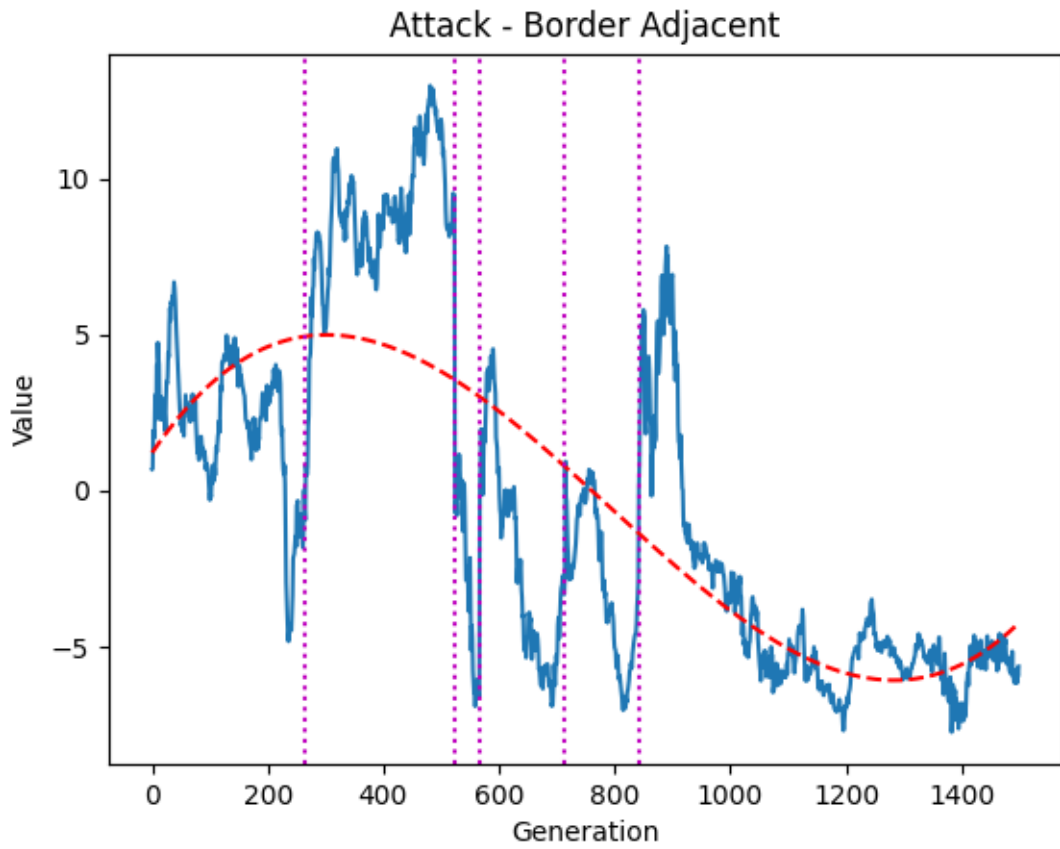


Figure 21. Graph of how the "Border Adjacent" characteristic from the "Attack" category changed over time.

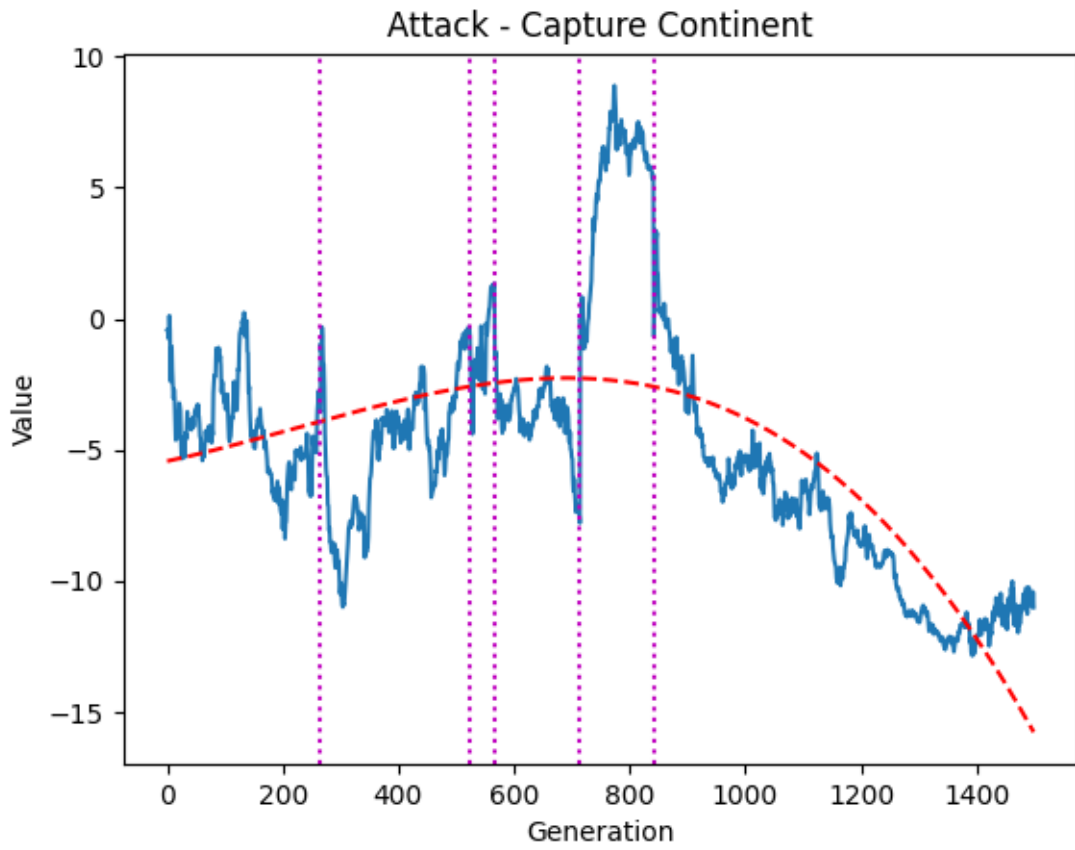


Figure 22. Graph of how the "Capture Continent" characteristic from the "Attack" category changed over time.

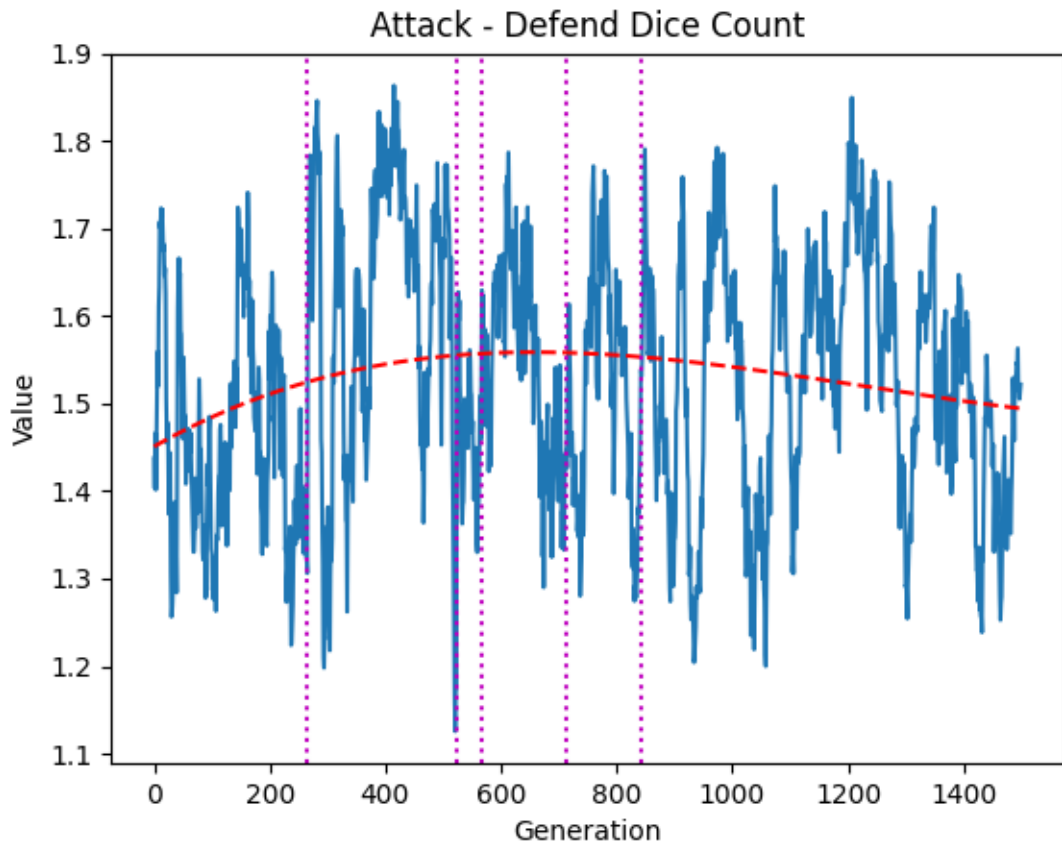


Figure 23. Graph of how the "Defend Dice Count" characteristic from the "Attack" category changed over time.

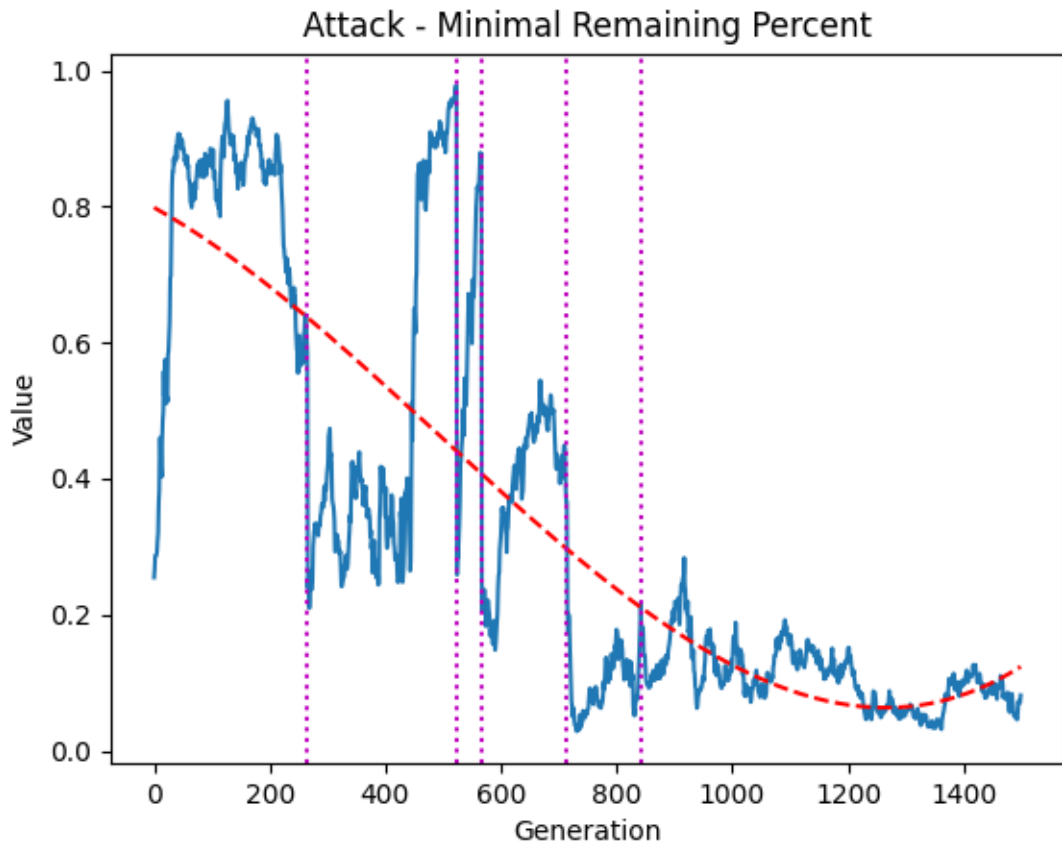


Figure 24. Graph of how the "Minimal Remaining Percent" characteristic from the "Attack" category changed over time.

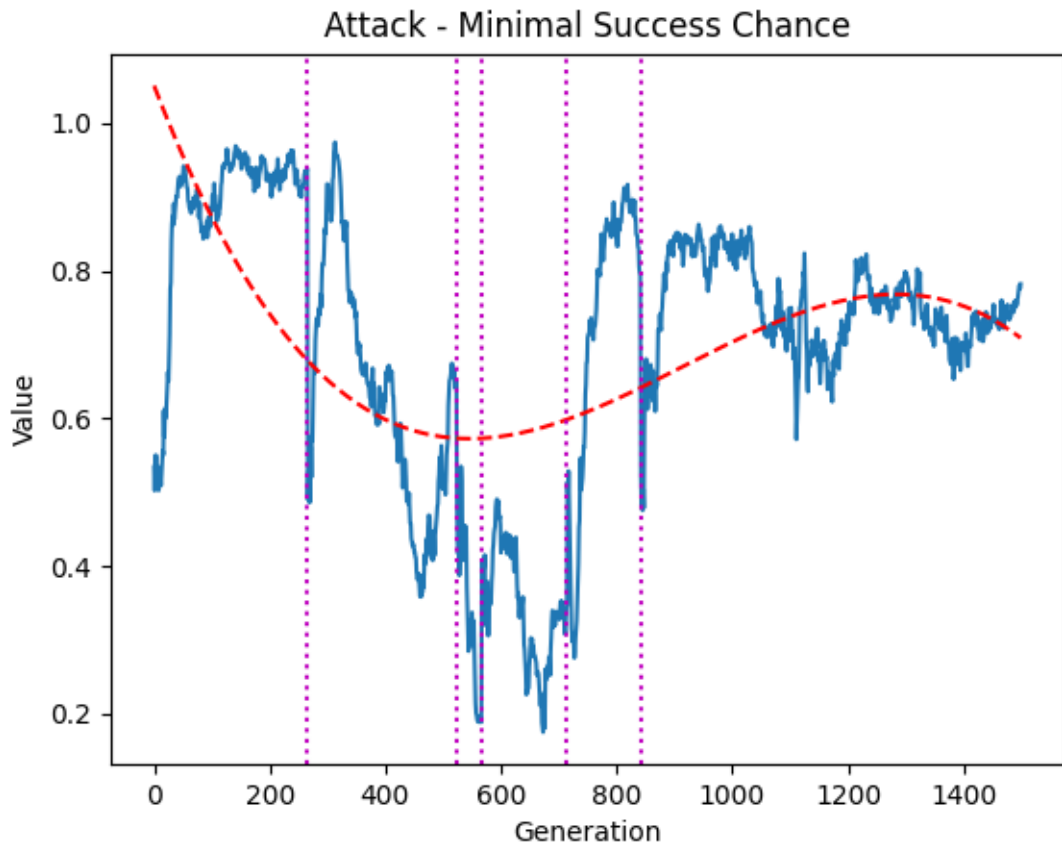


Figure 25. Graph of how the "Minimal Success Chance" characteristic from the "Attack" category changed over time.

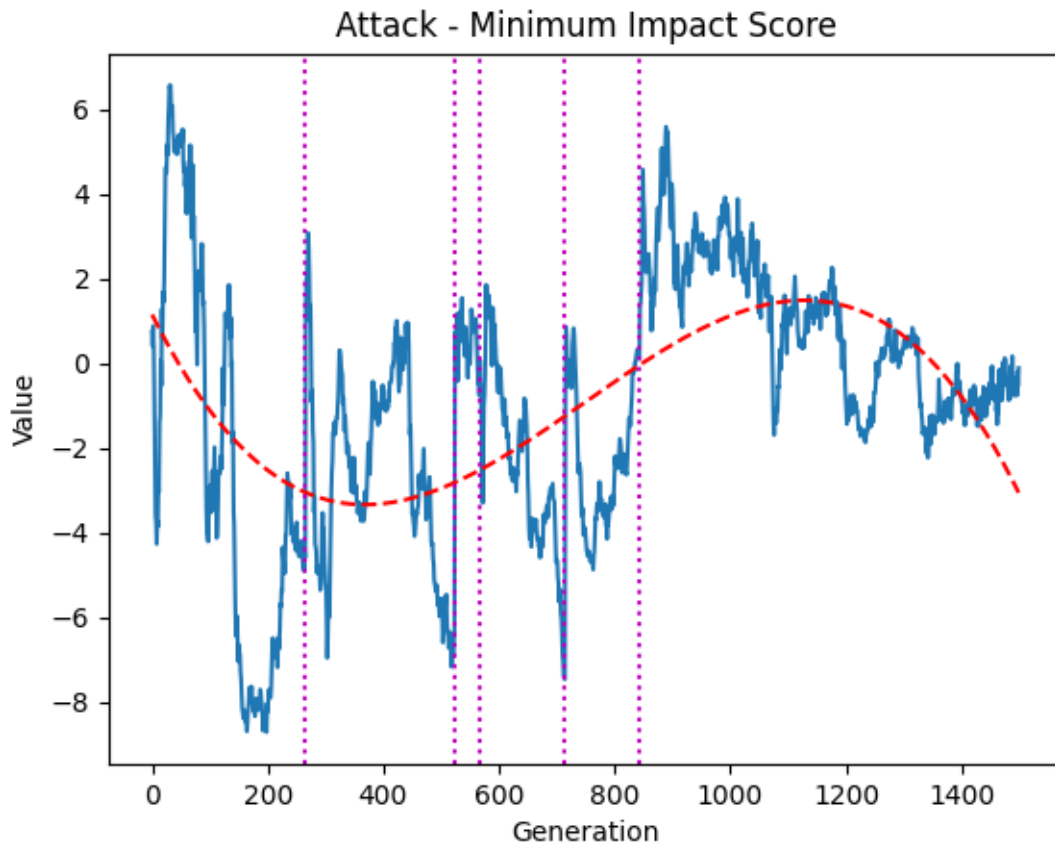


Figure 26. Graph of how the "Minimum Impact Score" characteristic from the "Attack" category changed over time.

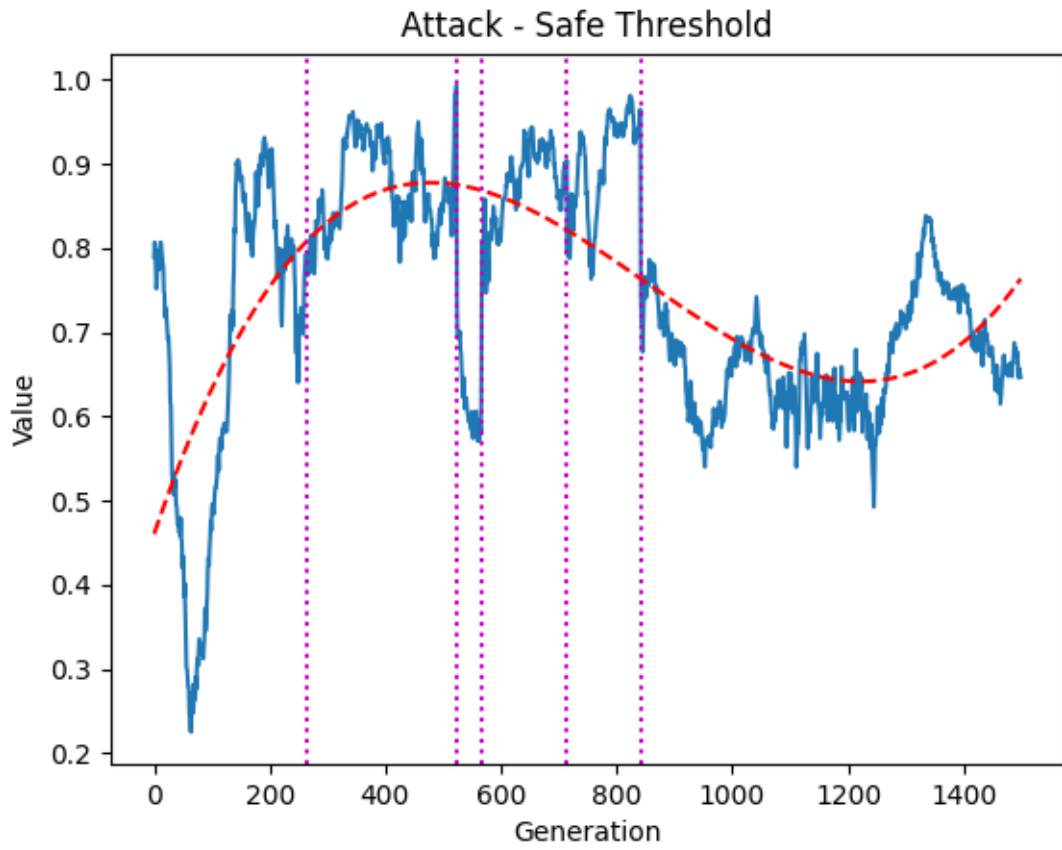


Figure 27. Graph of how the "Safe Threshold" characteristic from the "Attack" category changed over time.

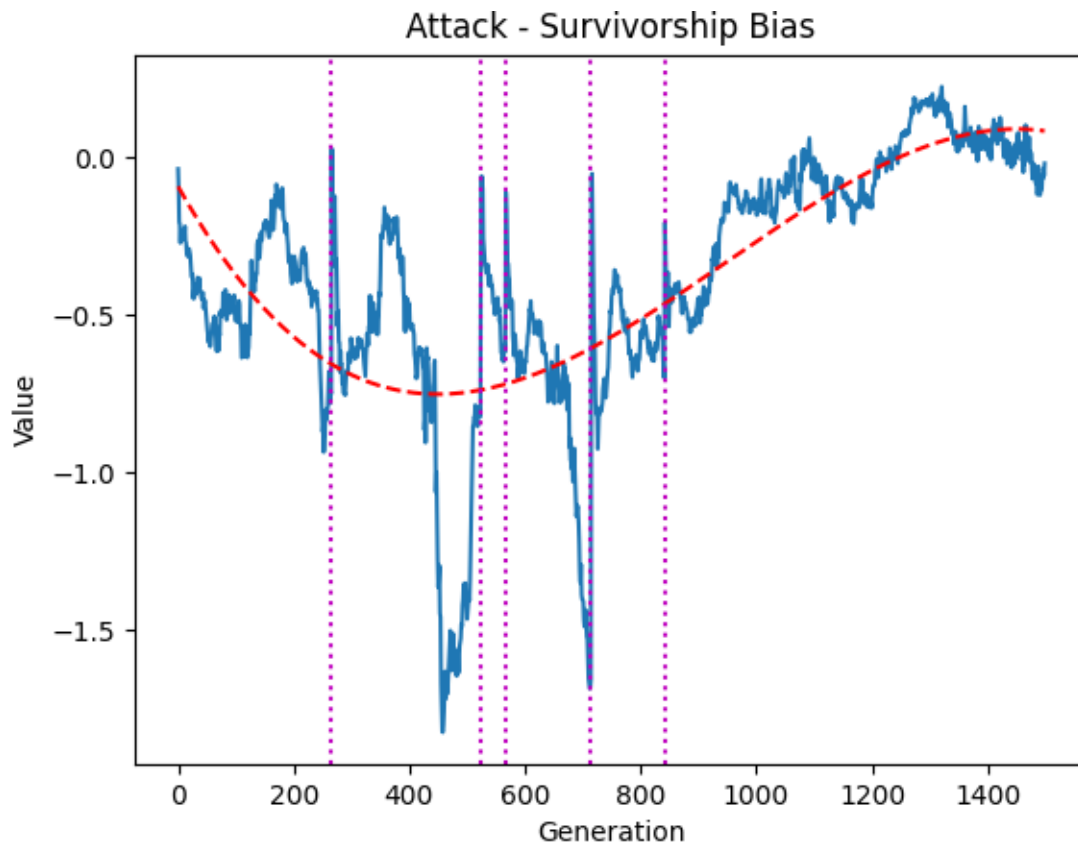


Figure 28. Graph of how the "Survivorship Bias" characteristic from the "Attack" category changed over time.

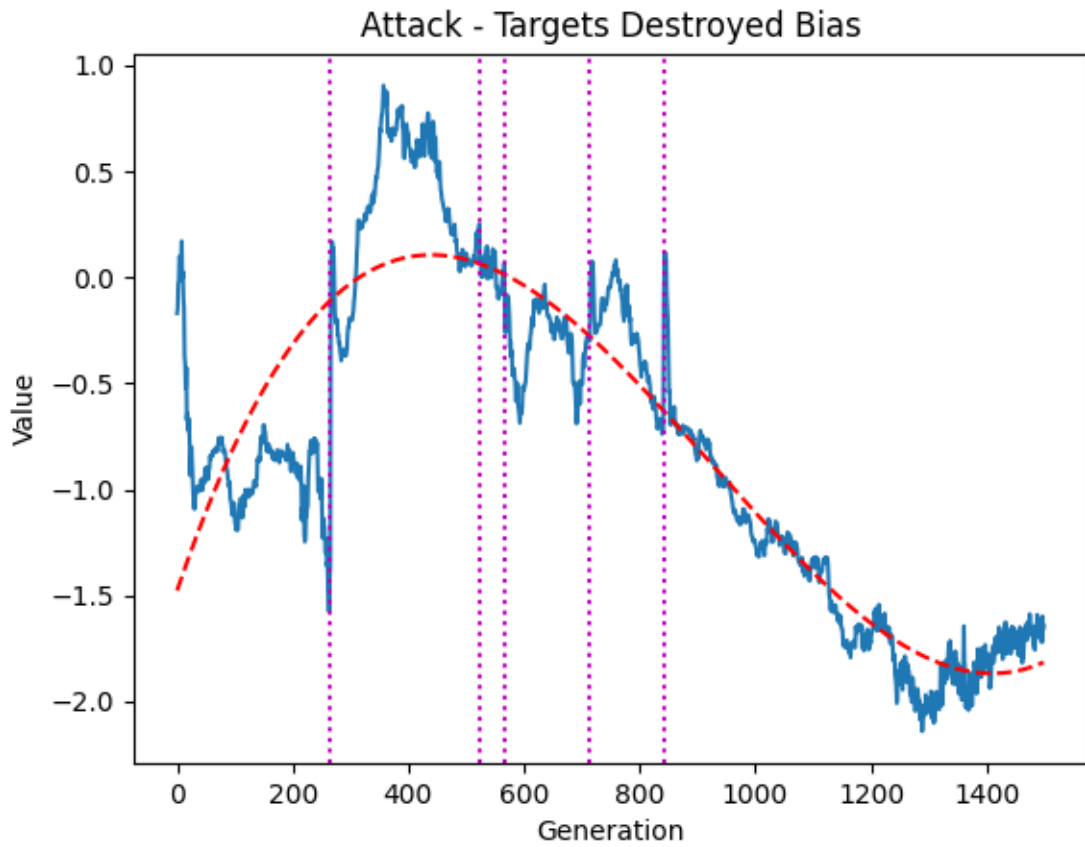


Figure 29. Graph of how the "Targets Destroyed Bias" characteristic from the "Attack" category changed over time.

Attack Continent Category

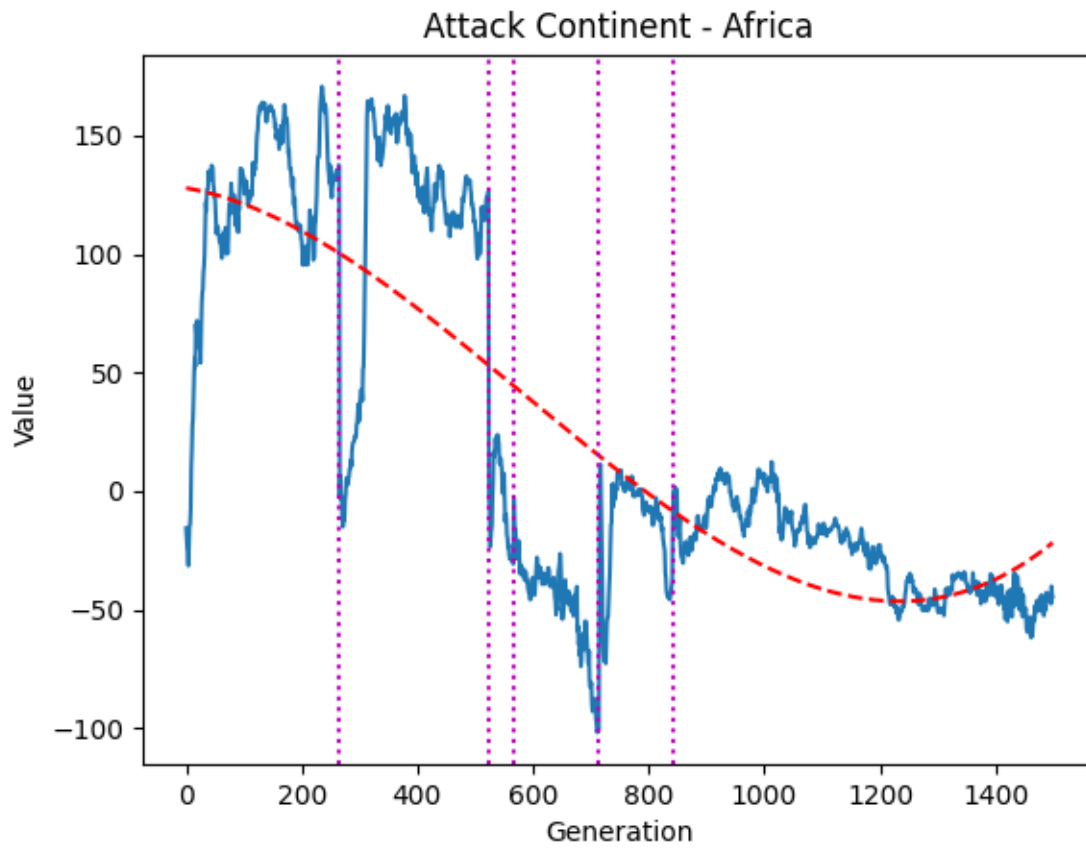


Figure 30. Graph of how the "Africa" characteristic from the "Attack Continent" category changed over time.

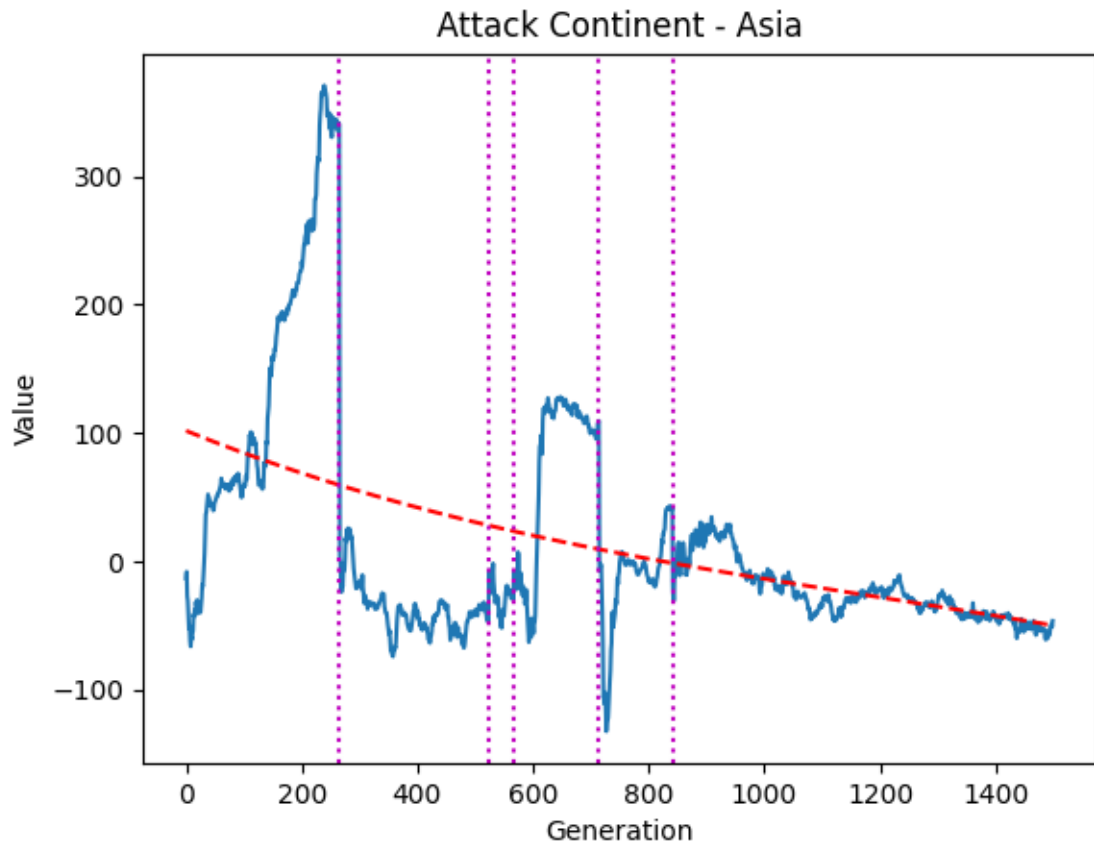


Figure 31. Graph of how the "Asia" characteristic from the "Attack Continent" category changed over time.

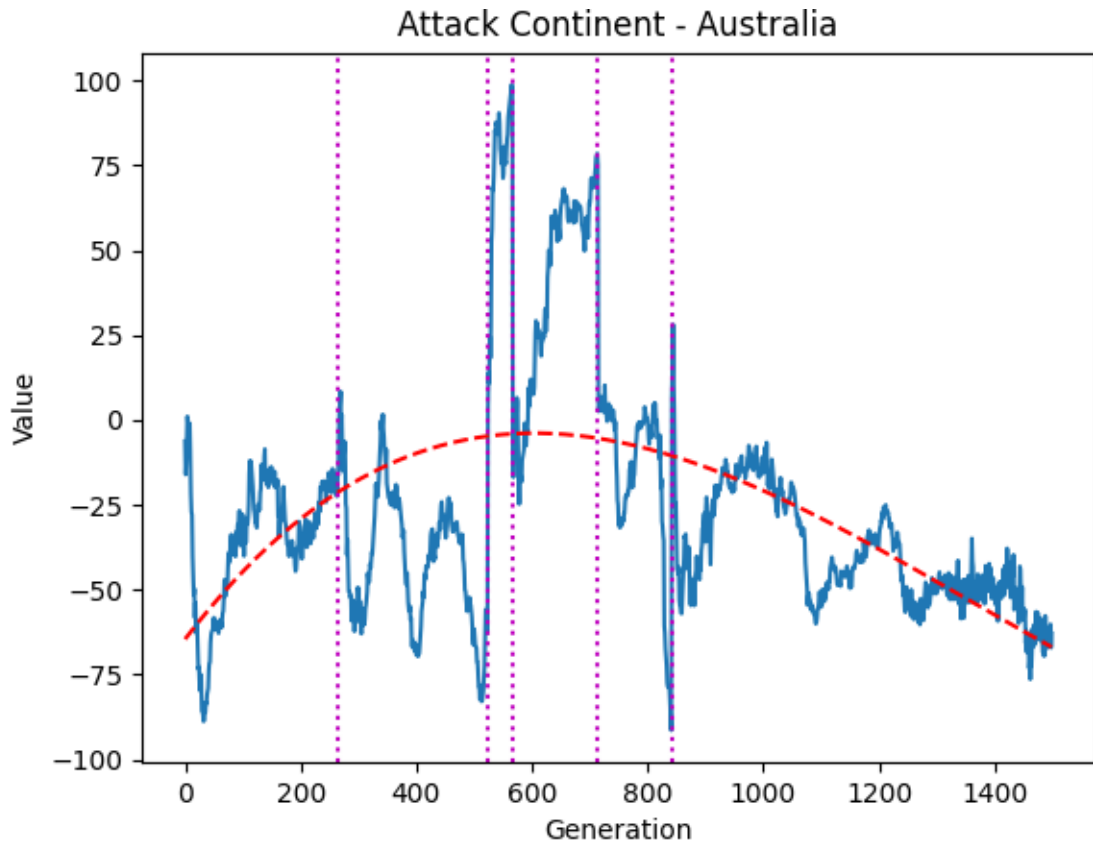


Figure 32. Graph of how the "Australia" characteristic from the "Attack Continent" category changed over time.

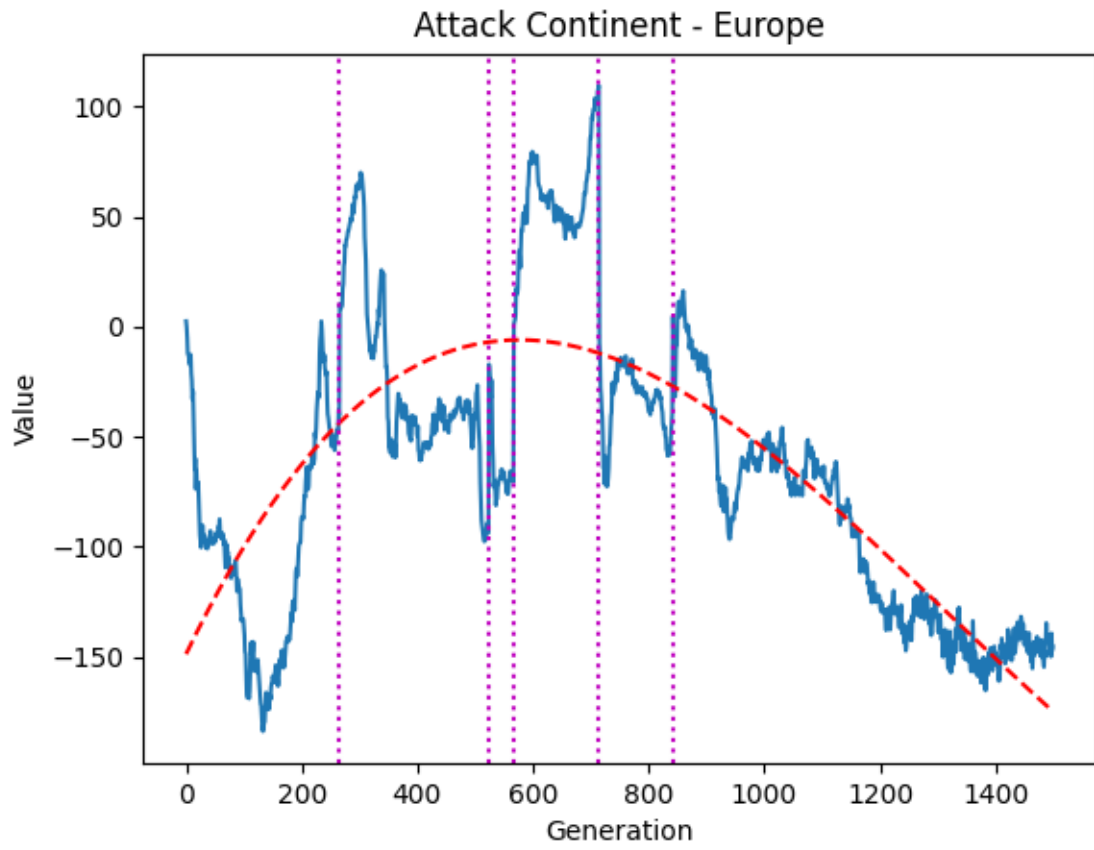


Figure 33. Graph of how the "Europe" characteristic from the "Attack Continent" category changed over time.

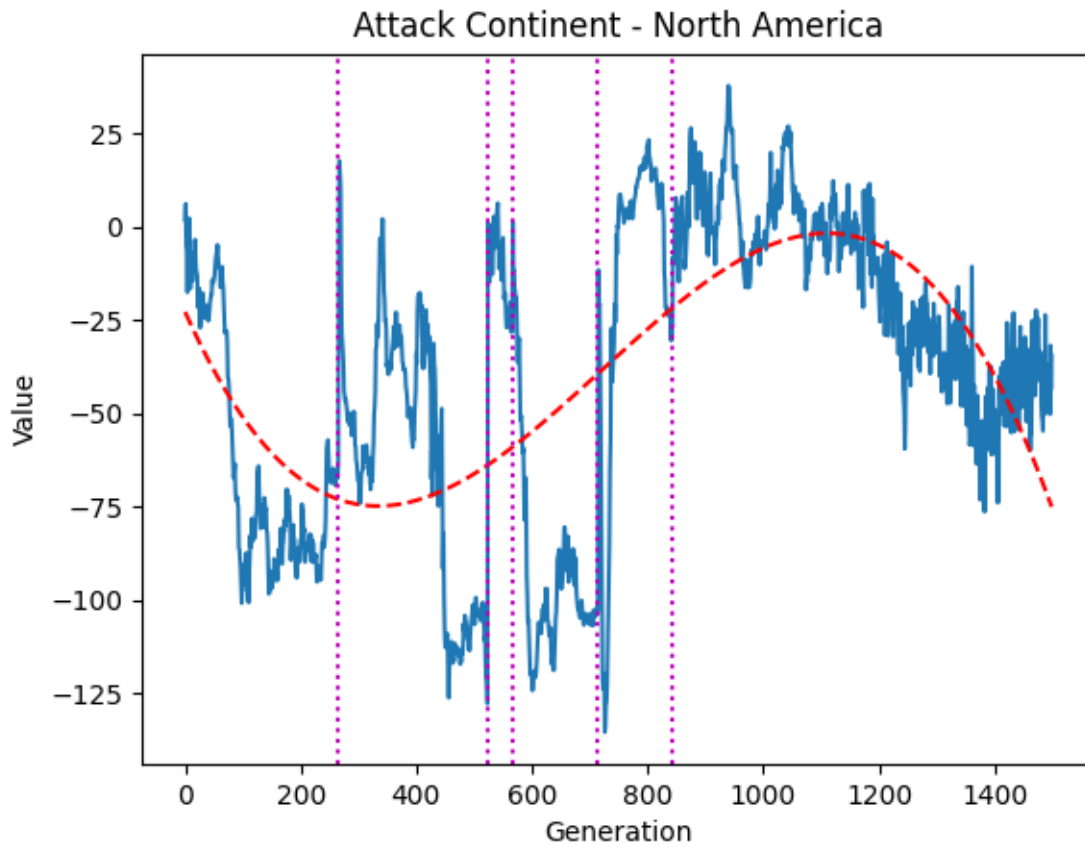


Figure 34. Graph of how the "North America" characteristic from the "Attack Continent" category changed over time.

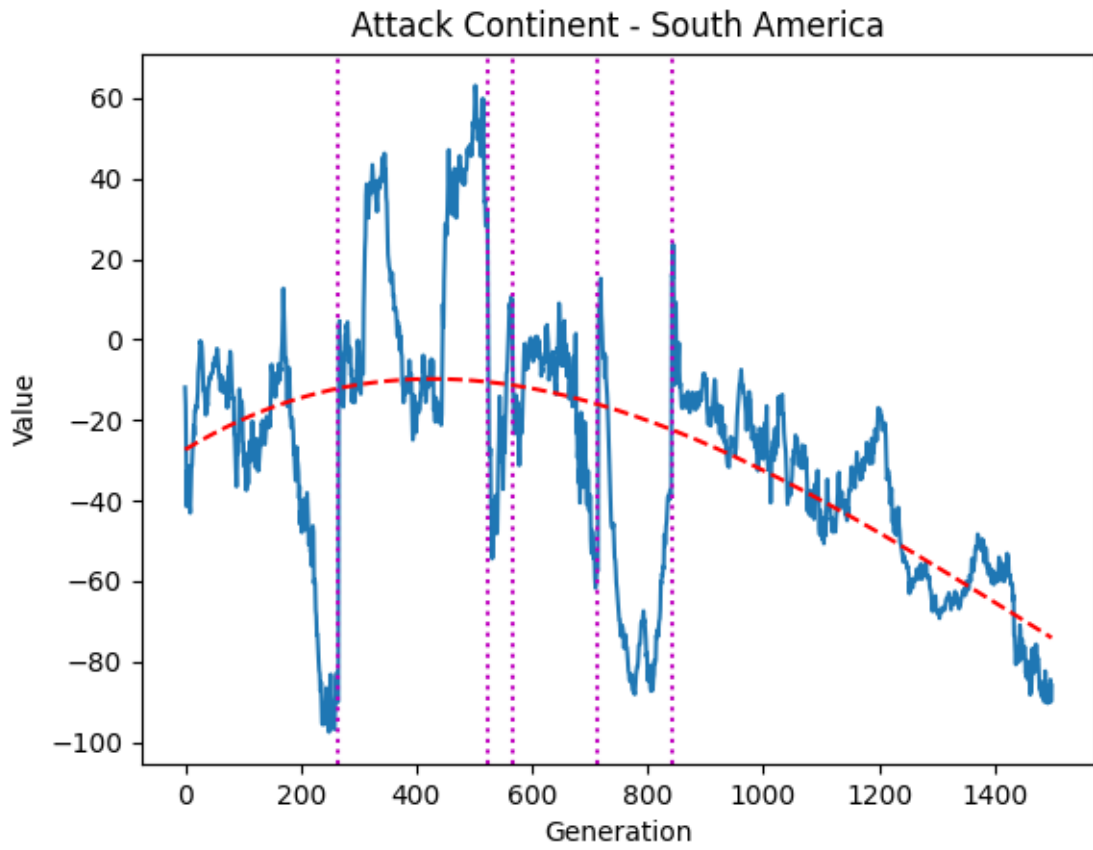


Figure 35. Graph of how the "South America" characteristic from the "Attack Continent" category changed over time.

Movement Category

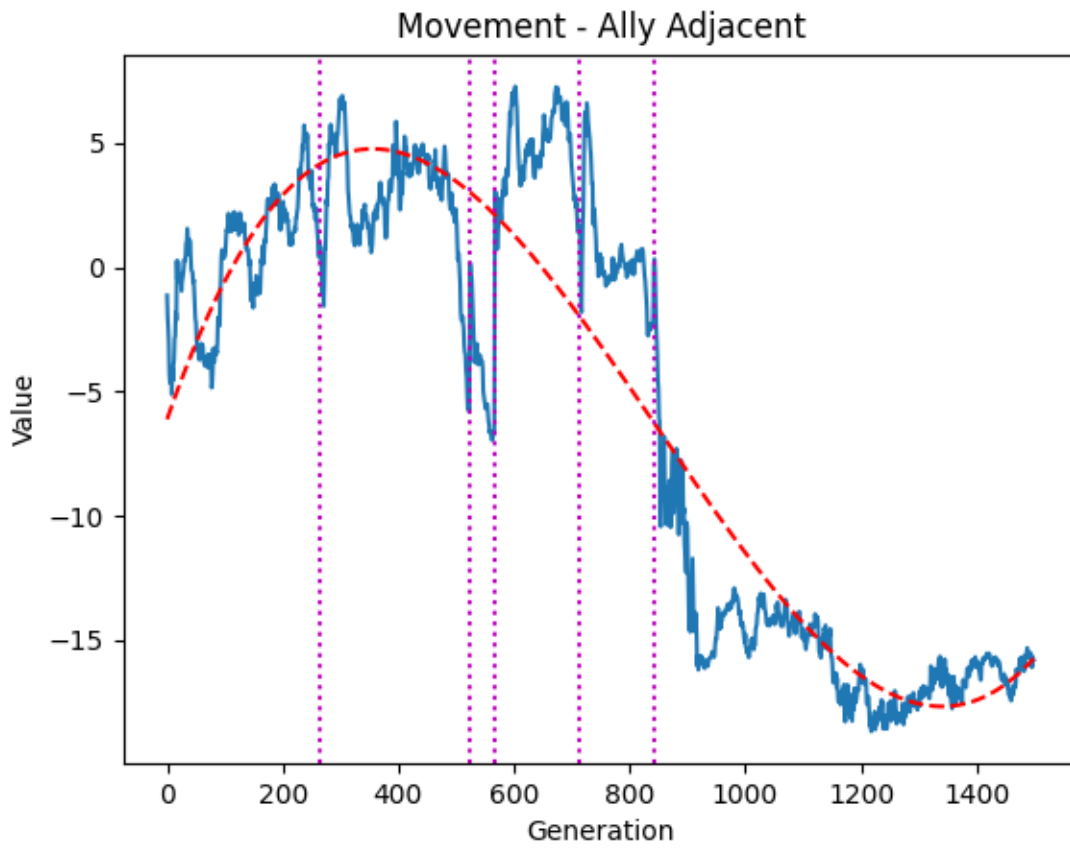


Figure 36. Graph of how the "Ally Adjacent" characteristic from the "Movement" category changed over time.

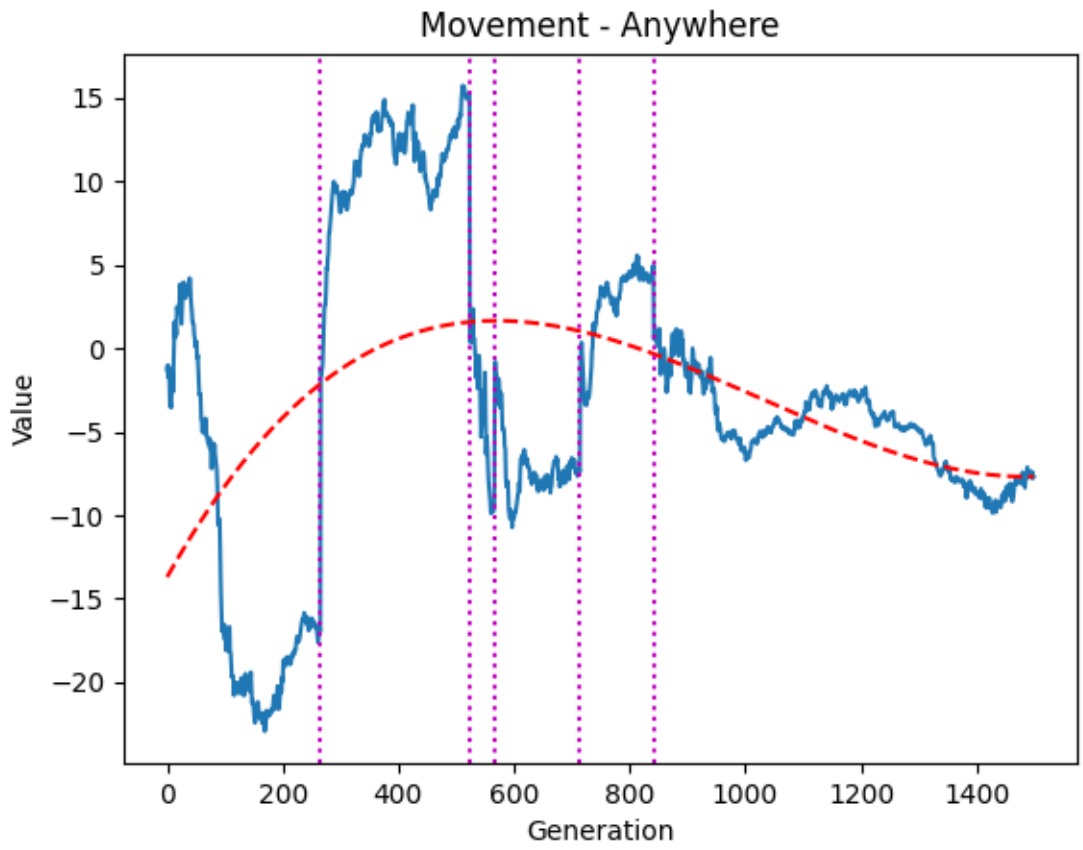


Figure 37. Graph of how the "Anywhere" characteristic from the "Movement" category changed over time.

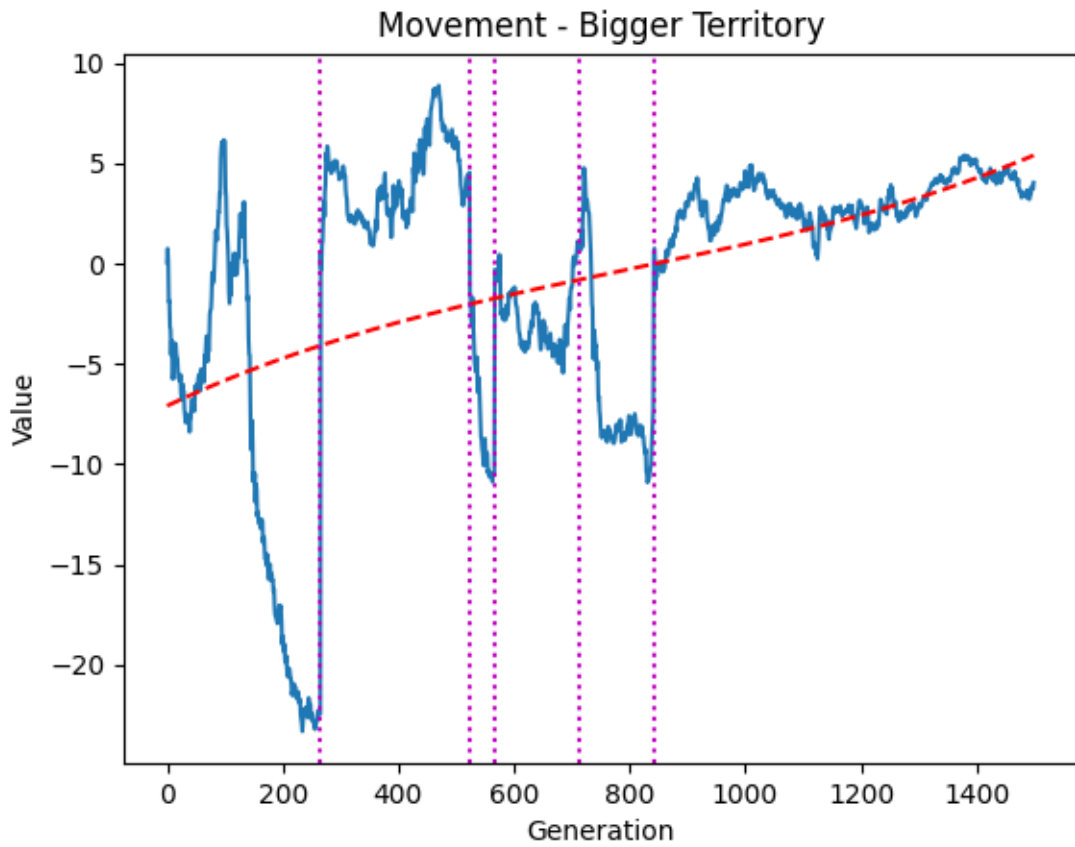


Figure 38. Graph of how the "Bigger Territory" characteristic from the "Movement" category changed over time.

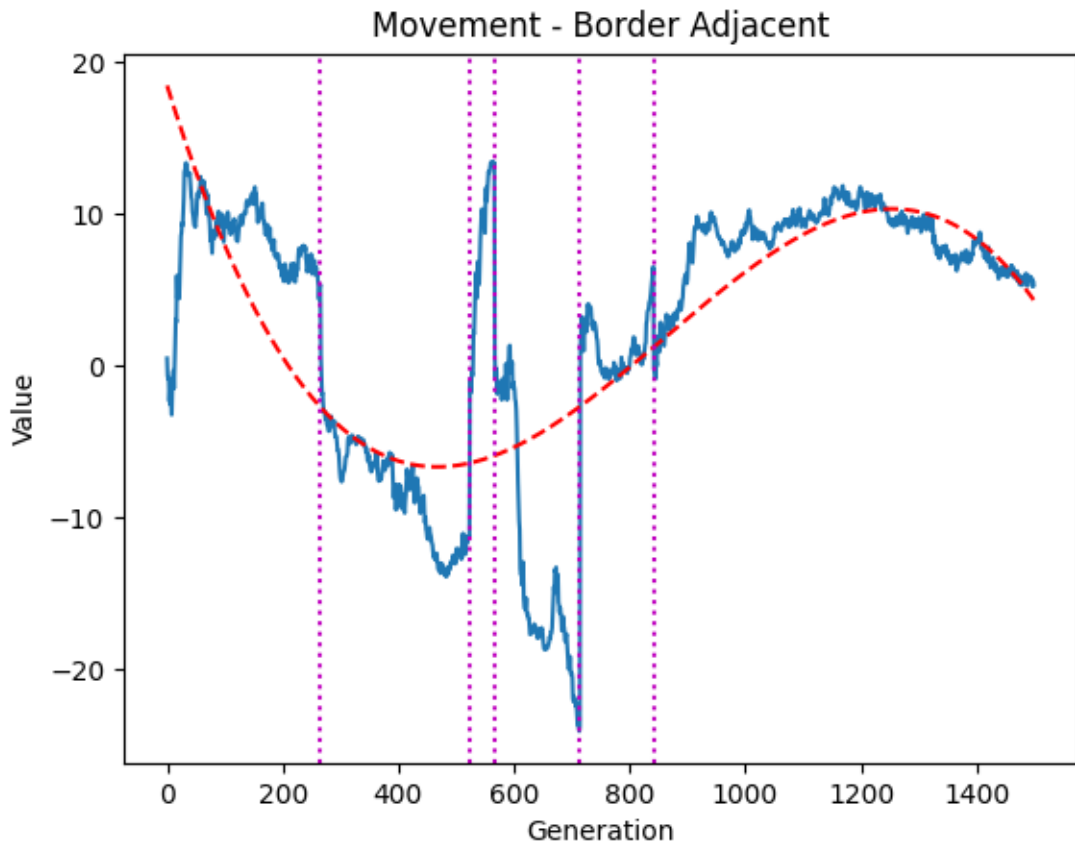


Figure 39. Graph of how the "Border Adjacent" characteristic from the "Movement" category changed over time.

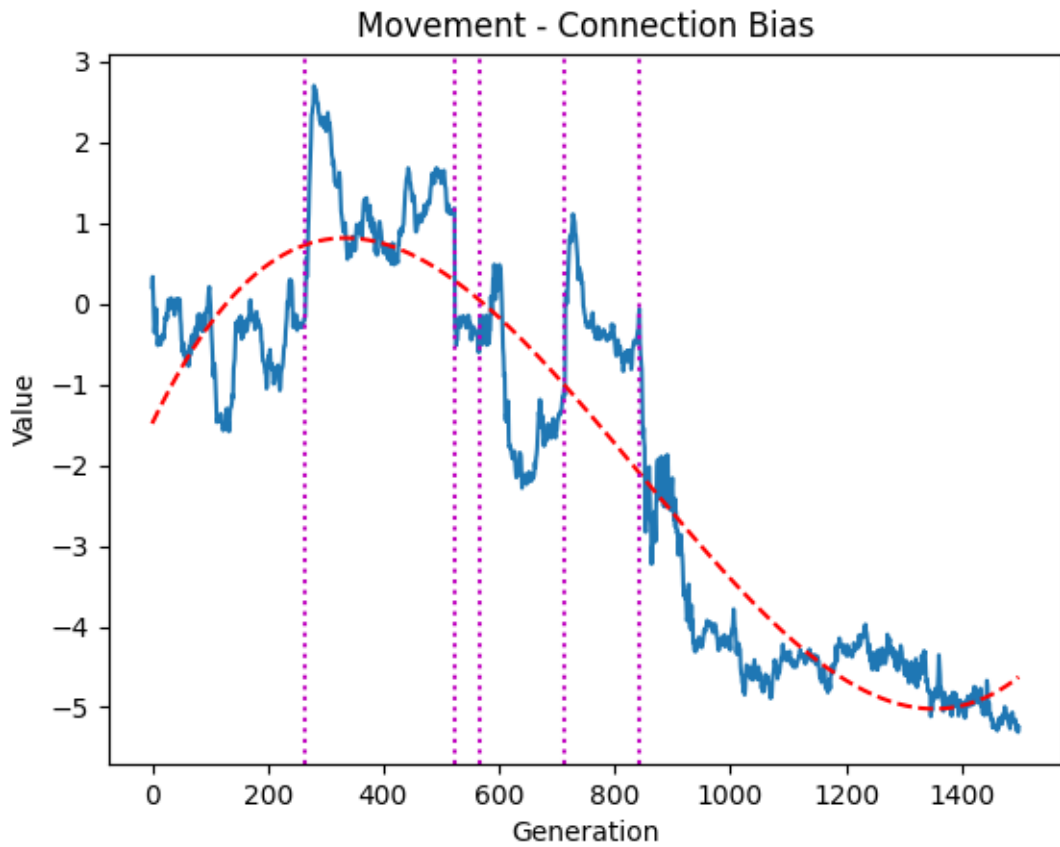


Figure 40. Graph of how the "Connection Bias" characteristic from the "Movement" category changed over time.

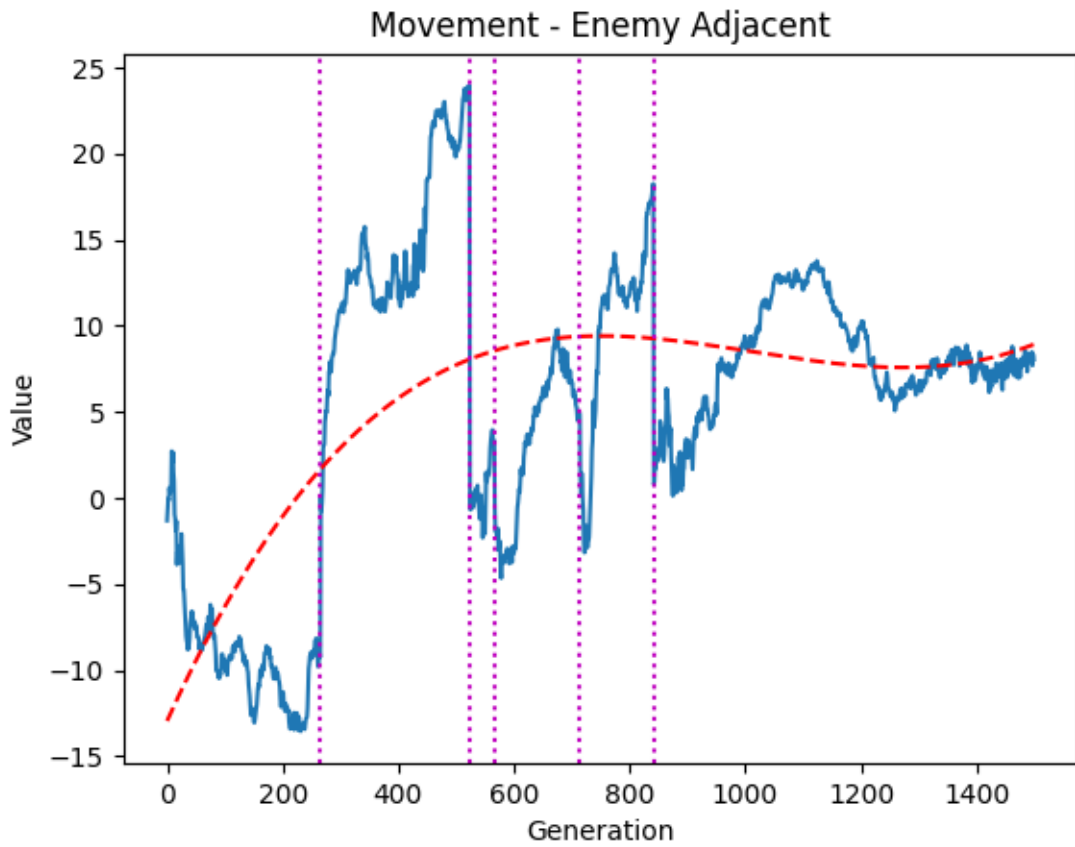


Figure 41. Graph of how the "Enemy Adjacent" characteristic from the "Movement" category changed over time.

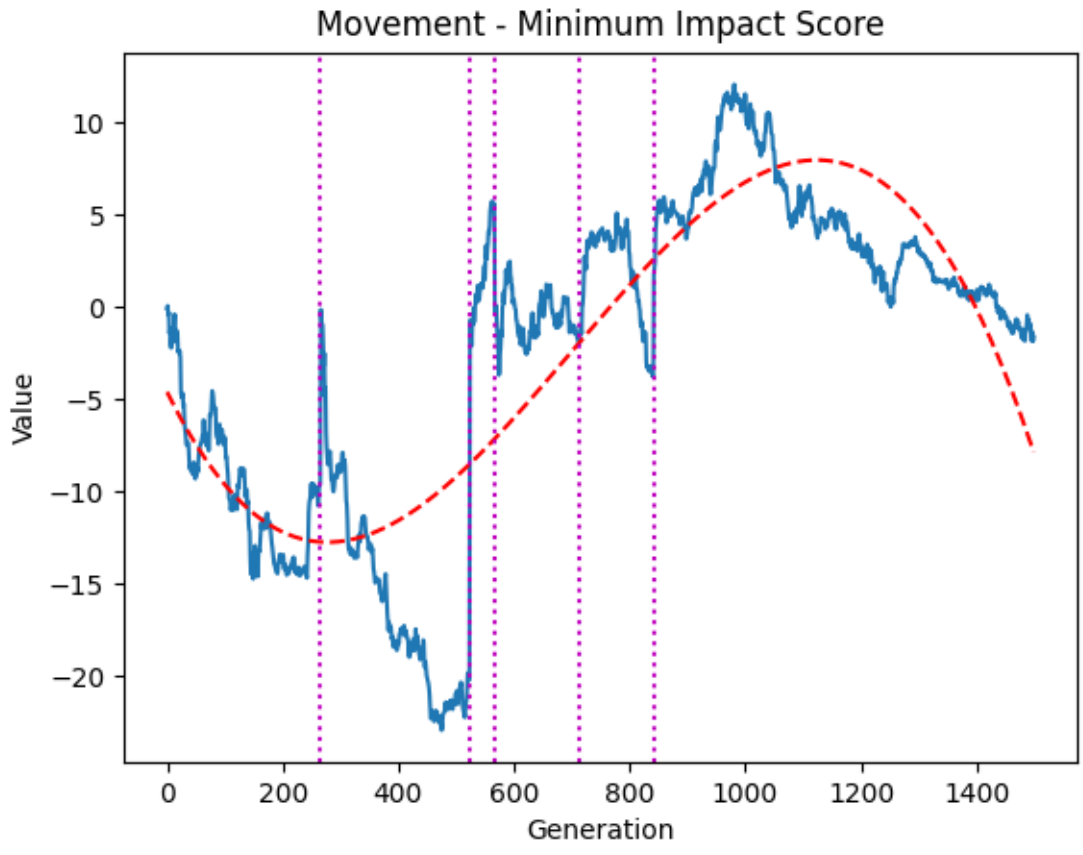


Figure 42. Graph of how the "Minimum Impact Score" characteristic from the "Movement" category changed over time.

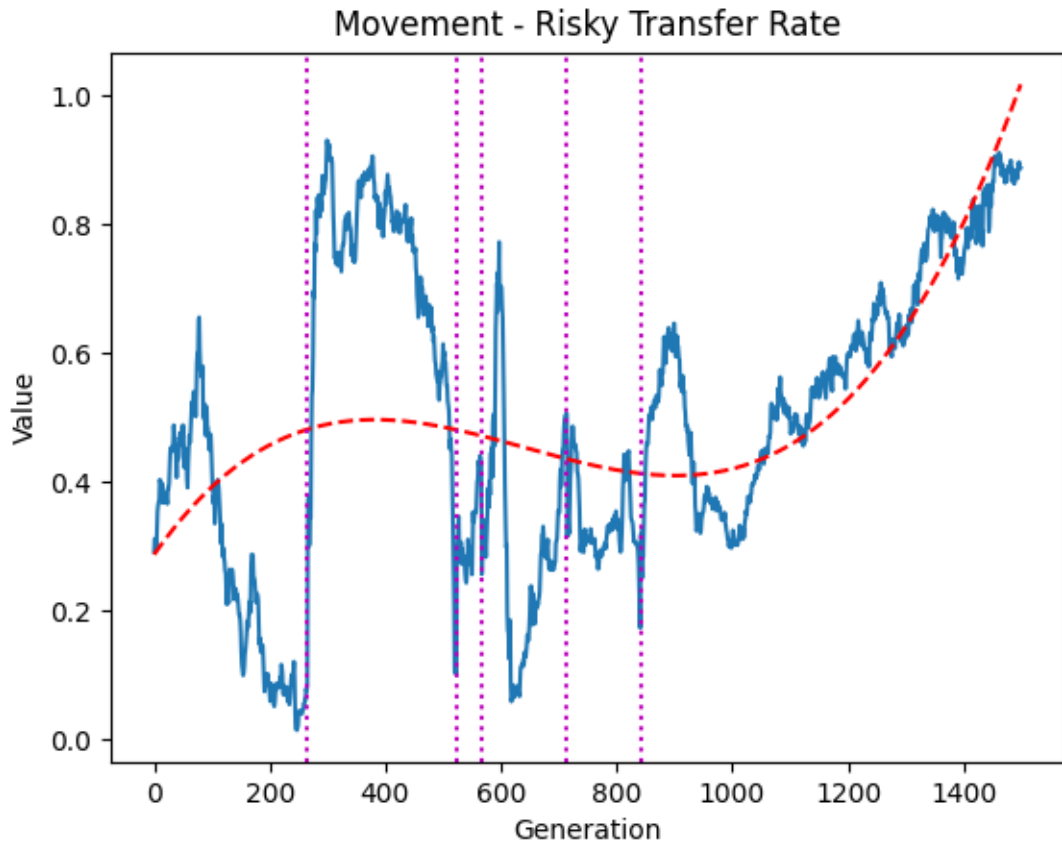


Figure 43. Graph of how the "Risky Transfer Rate" characteristic from the "Movement" category changed over time.

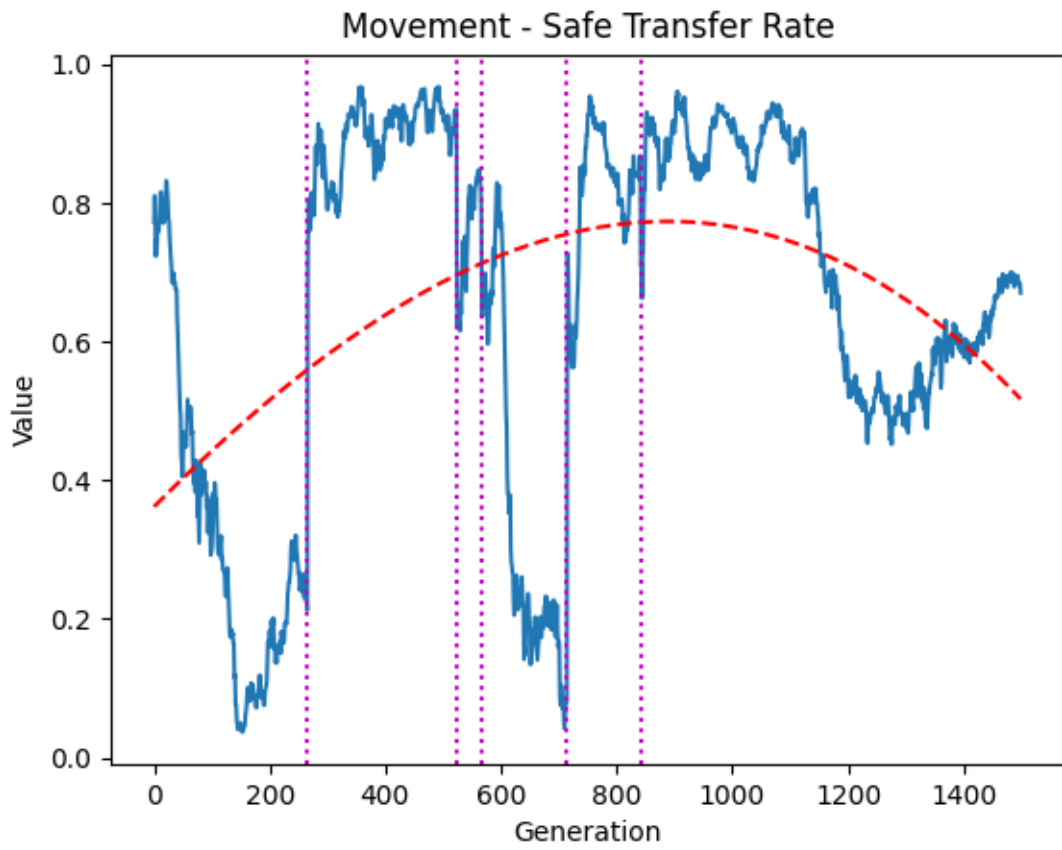


Figure 44. Graph of how the "Safe Transfer Rate" characteristic from the "Movement" category changed over time.

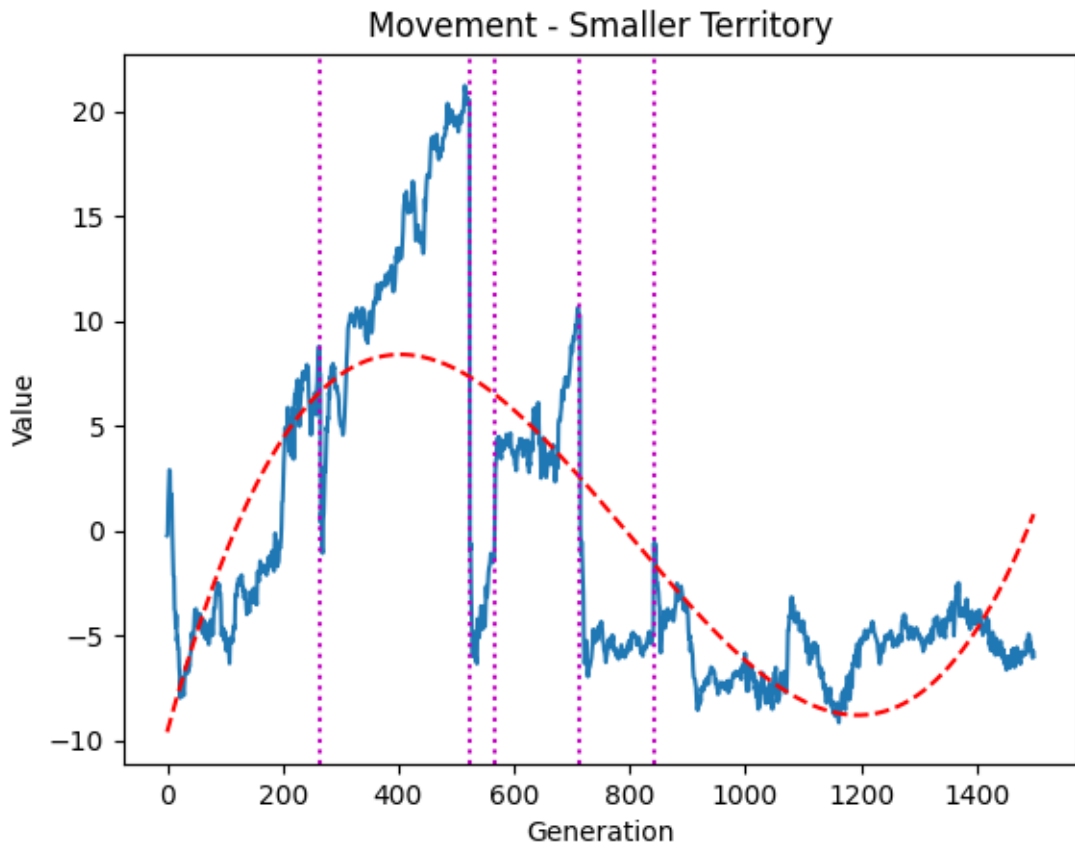


Figure 45. Graph of how the "Small Territory" characteristic from the "Movement" category changed over time.

Preference Category

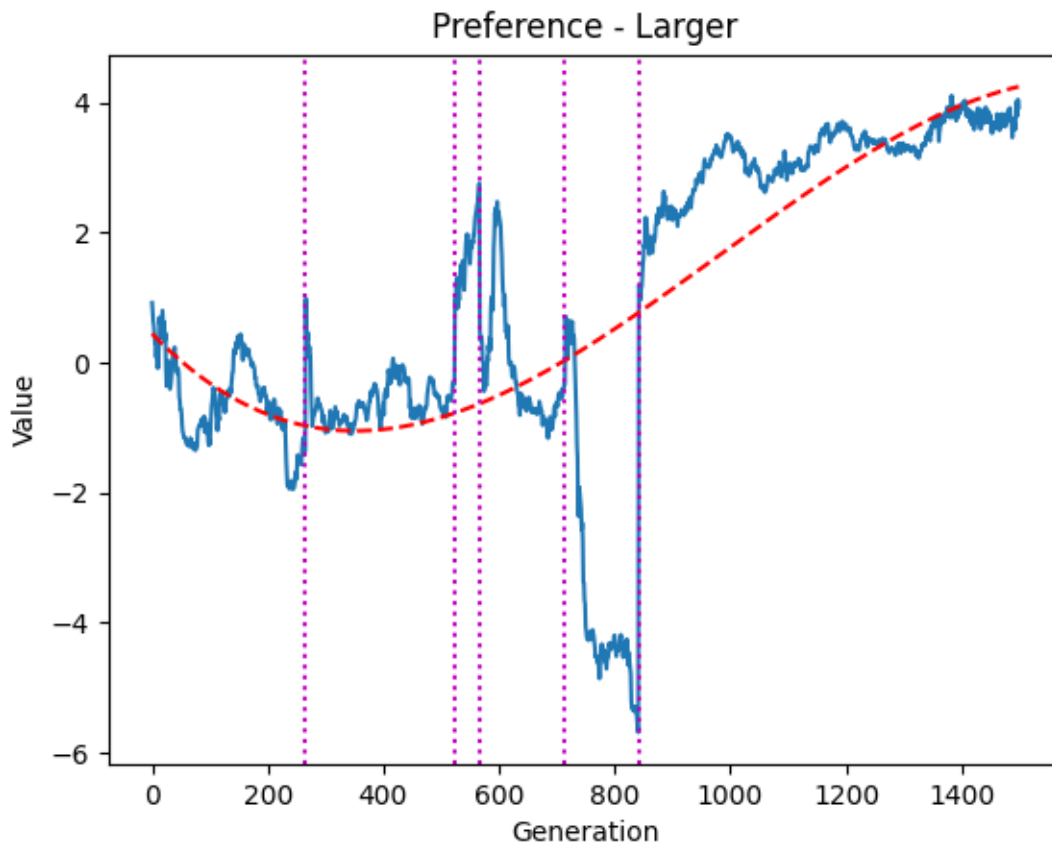


Figure 46. Graph of how the "Larger" characteristic from the "Preference" category changed over time.

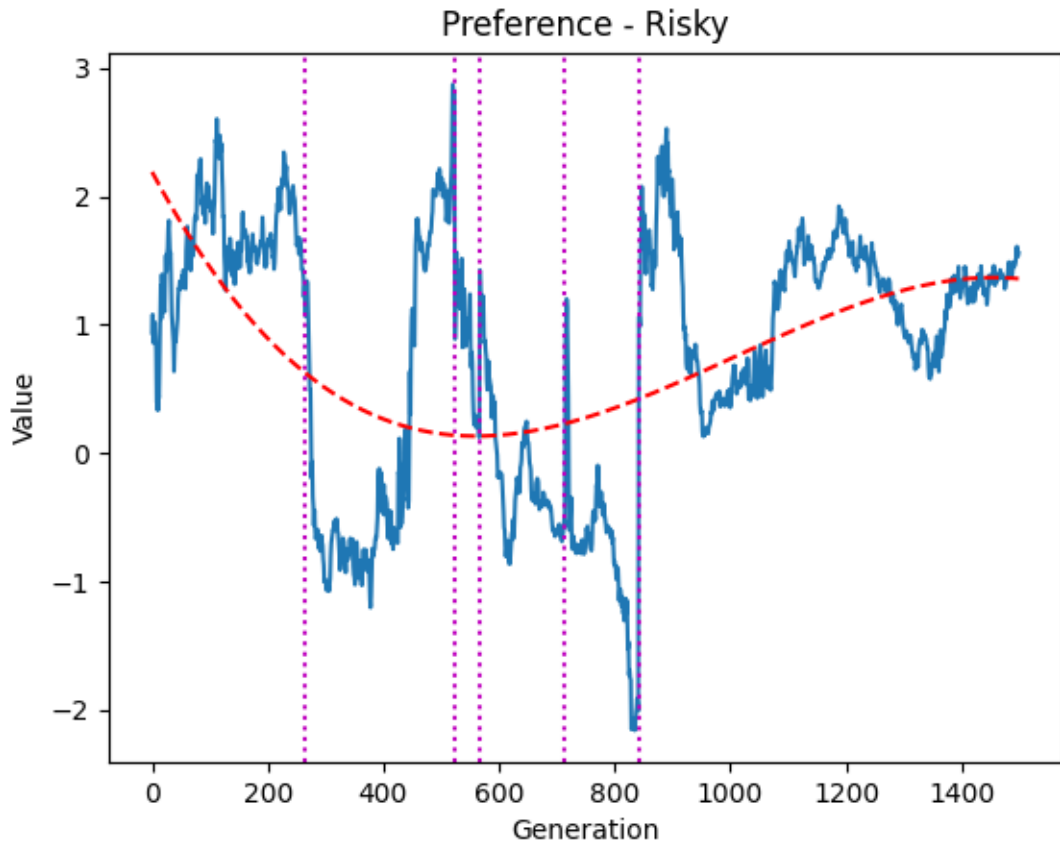


Figure 47. Graph of how the "Risky" characteristic from the "Preference" category changed over time.

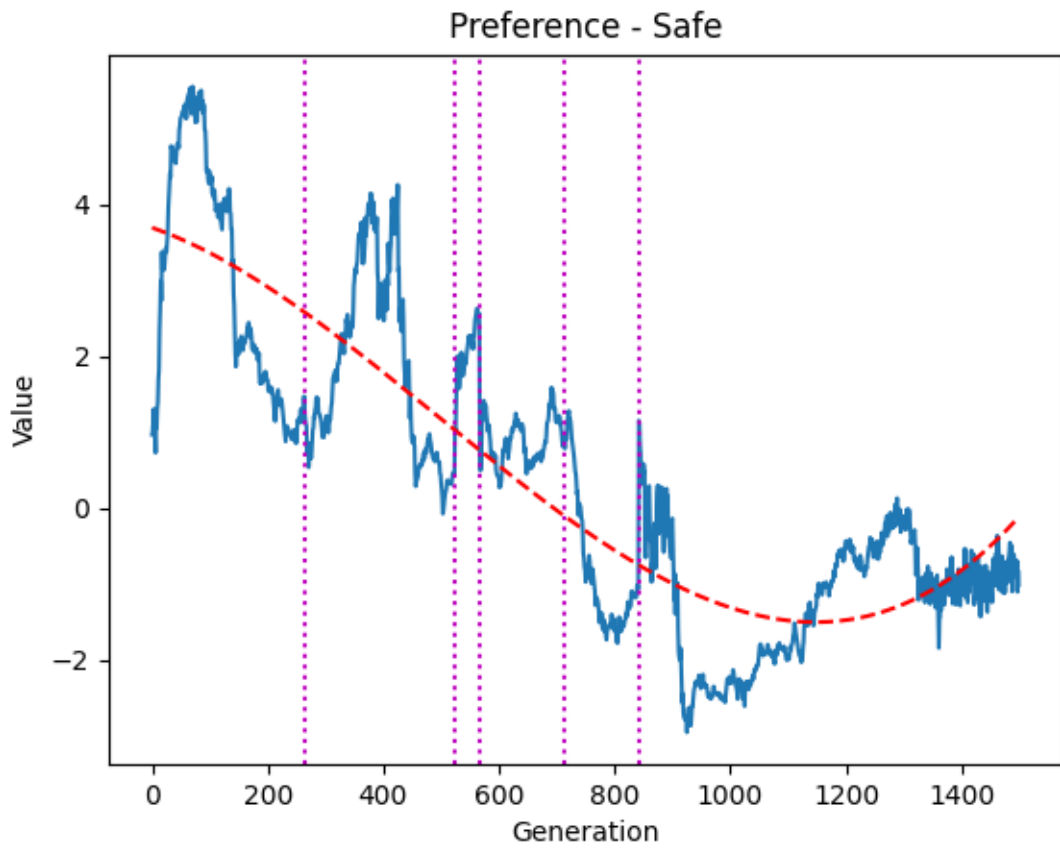


Figure 48. Graph of how the "Safe" characteristic from the "Preference" category changed over time.

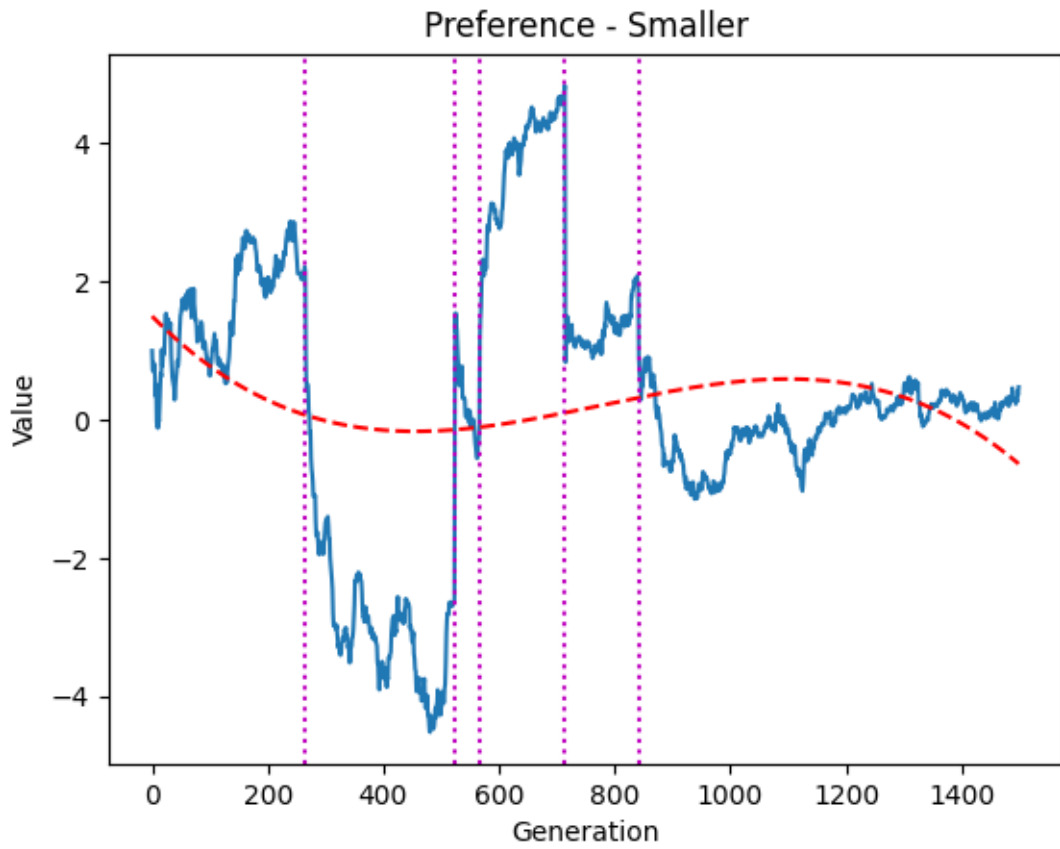


Figure 49. Graph of how the "Smaller" characteristic from the "Preference" category changed over time.

Appendix C: Dice Analysis Results

Maximum Attackers Strategy

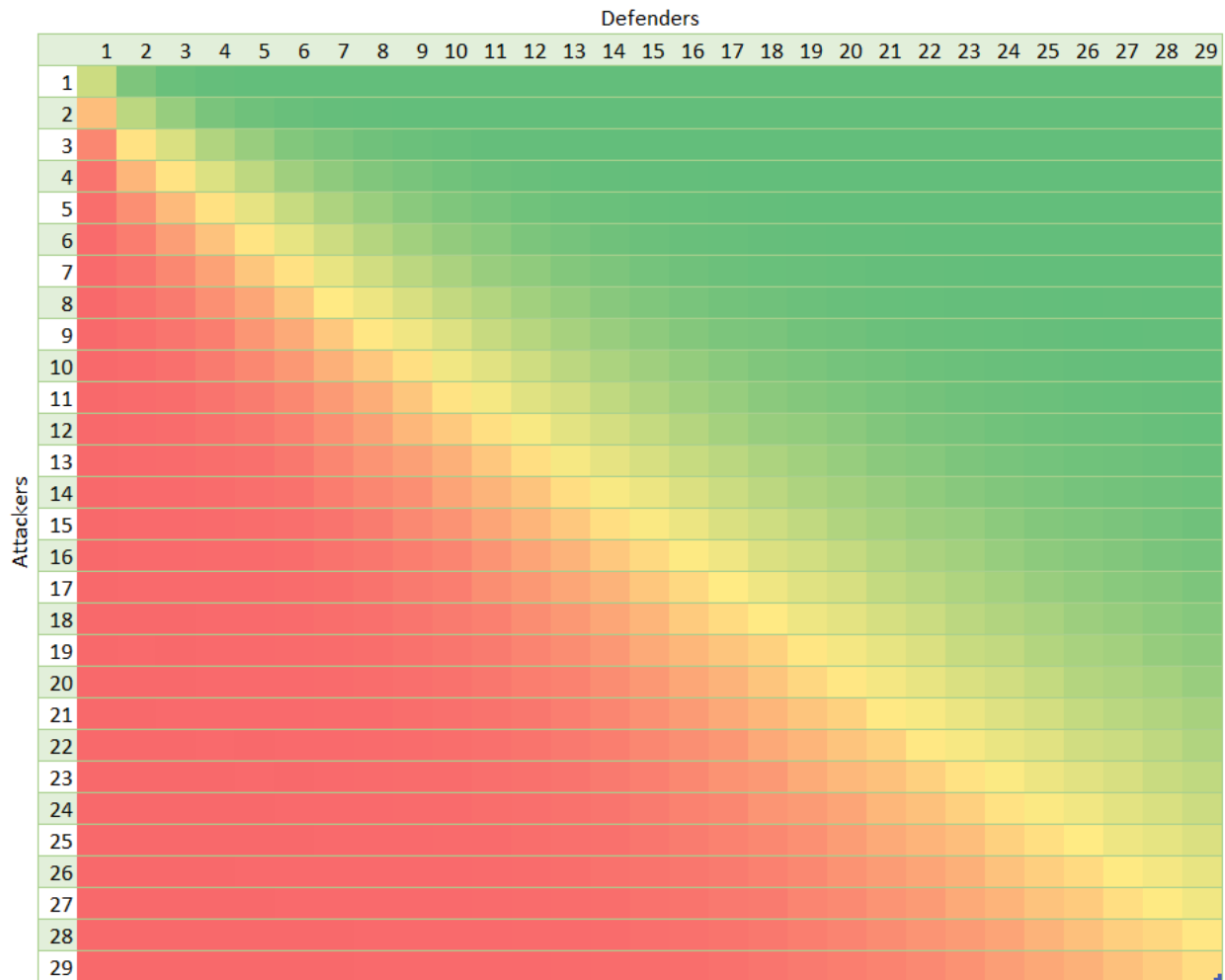


Figure 50. Heatmap of attack success chance using a maximum attackers strategy. The defender is using a maximum defenders strategy.

Percent Change Between Attacker Strategies

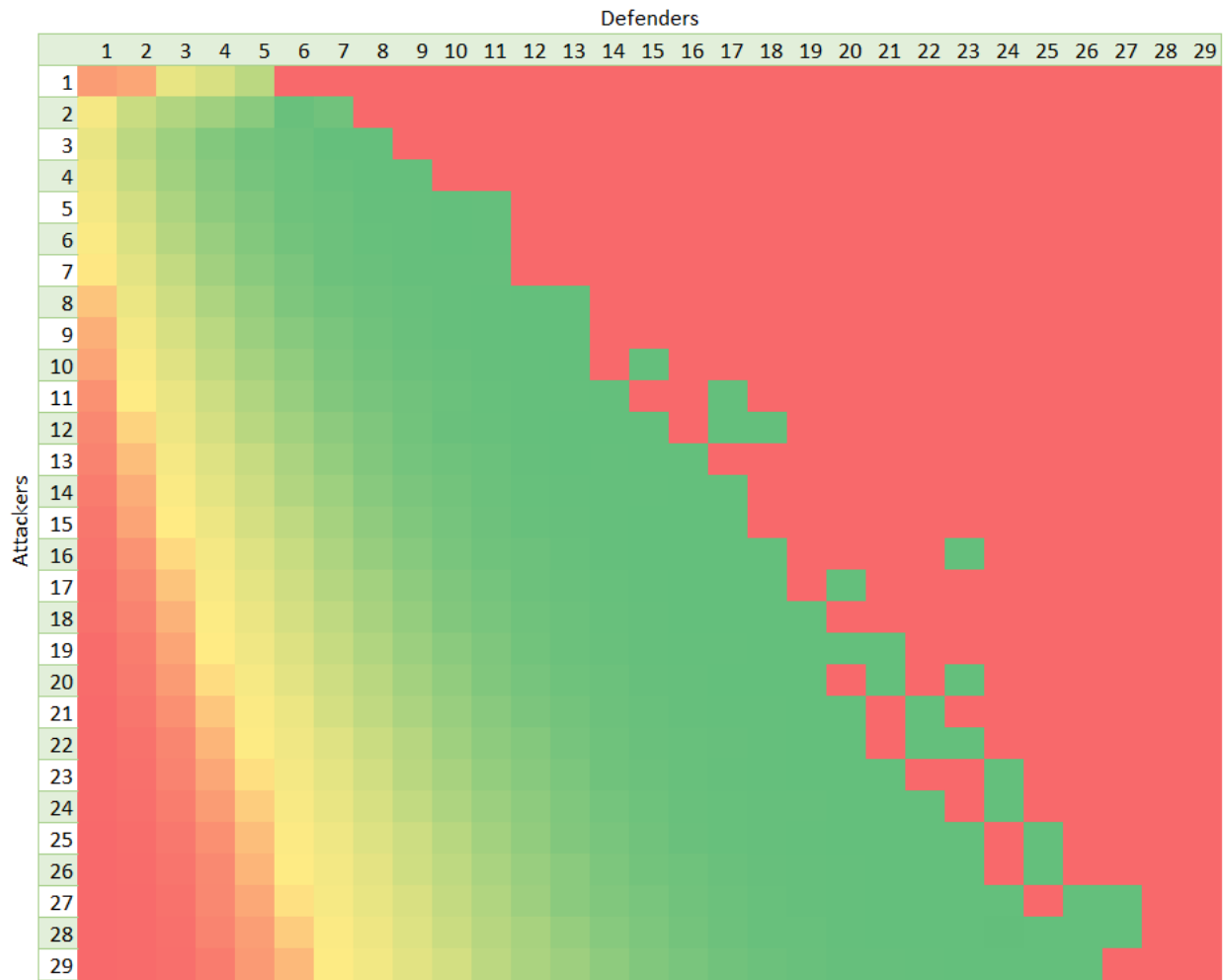


Figure 52. Heatmap of the difference between the two attack strategies. Red indicates no change.

Maximum Defenders Strategy

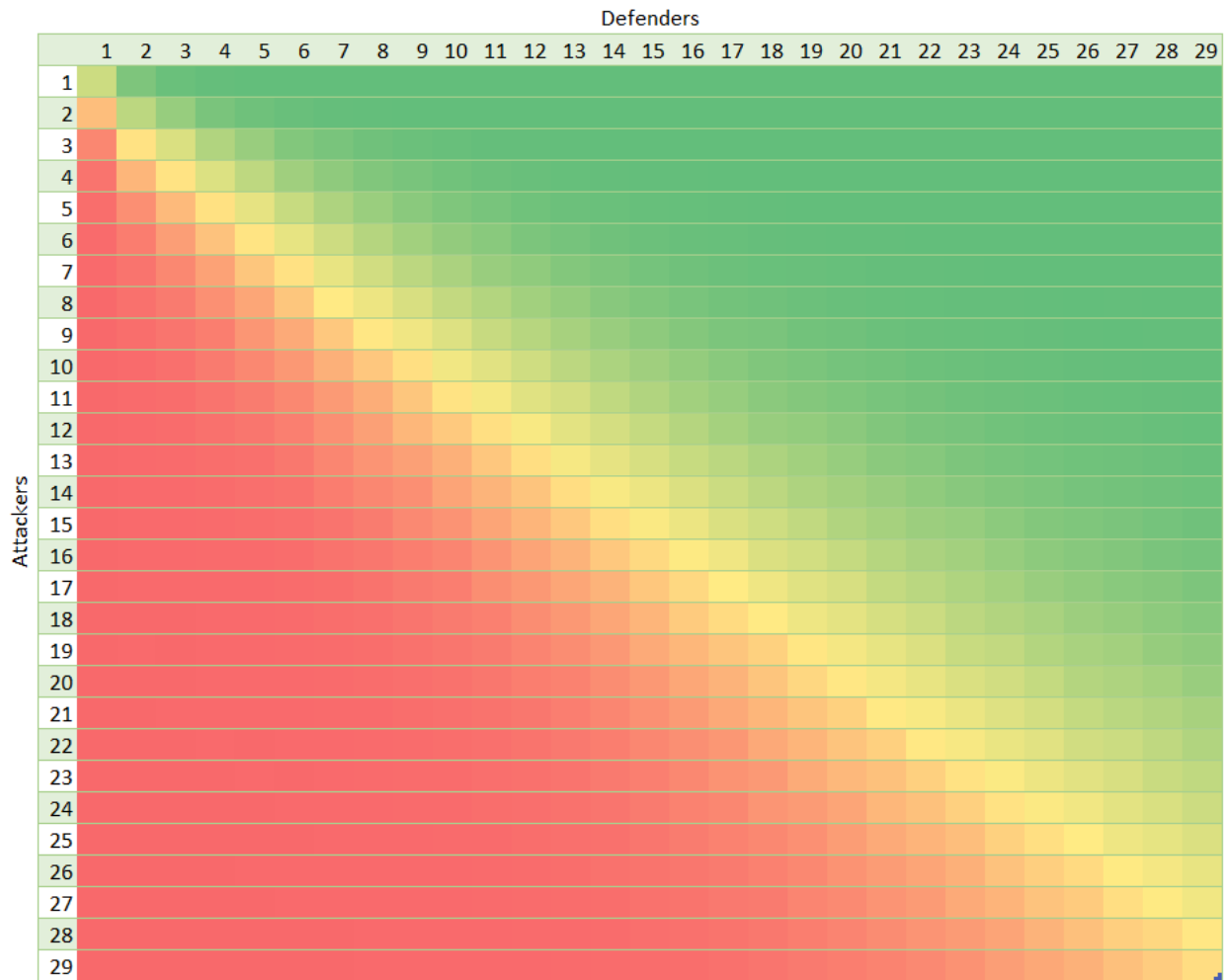


Figure 53. Heatmap of attack success chance using a maximum defenders strategy. The attacker is using a maximum attackers strategy.

Minimum Defenders Strategy

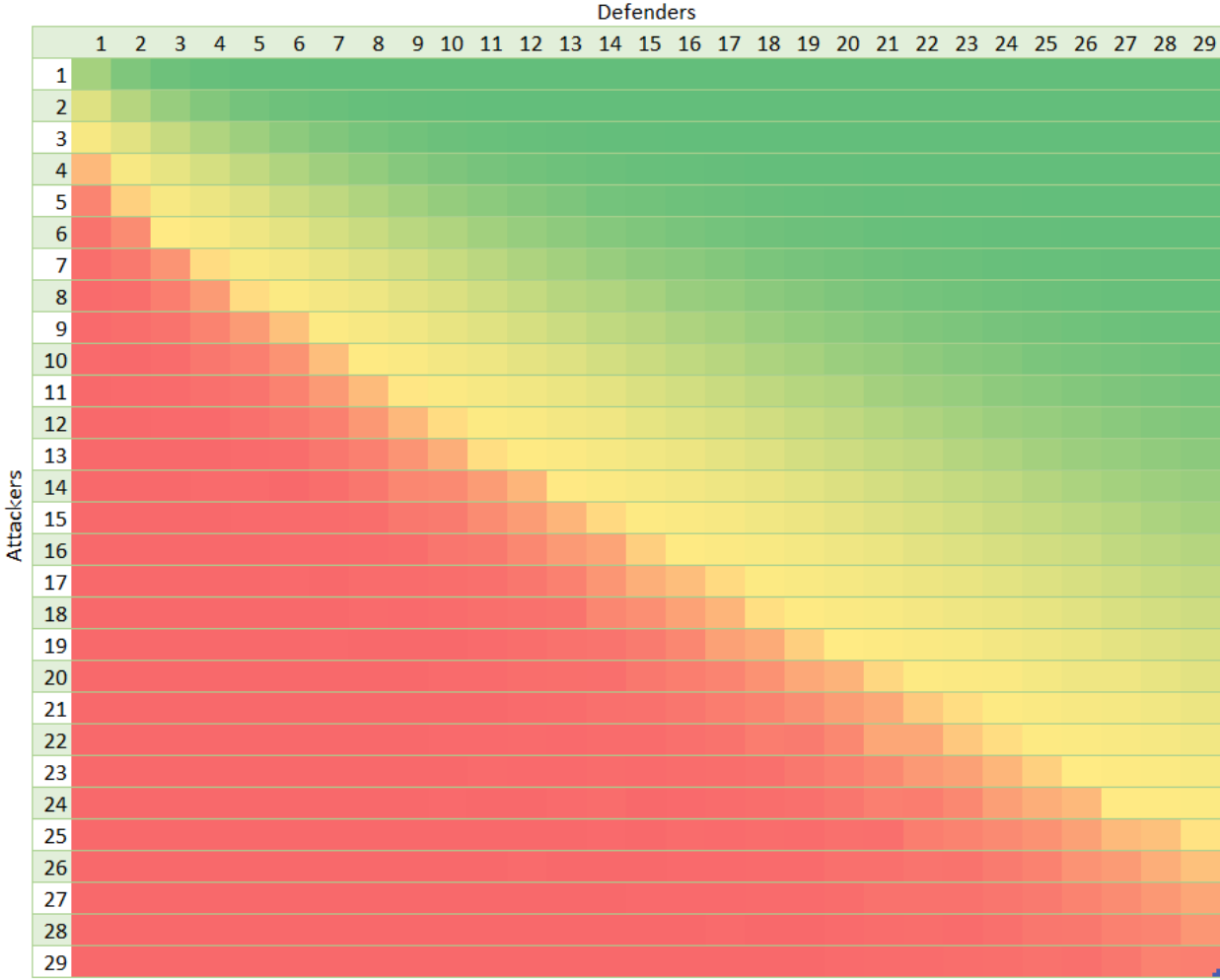


Figure 54. Heatmap of attack success chance using a minimum defenders strategy. The attacker is using a maximum attackers strategy.

Percent Change Between Defender Strategies

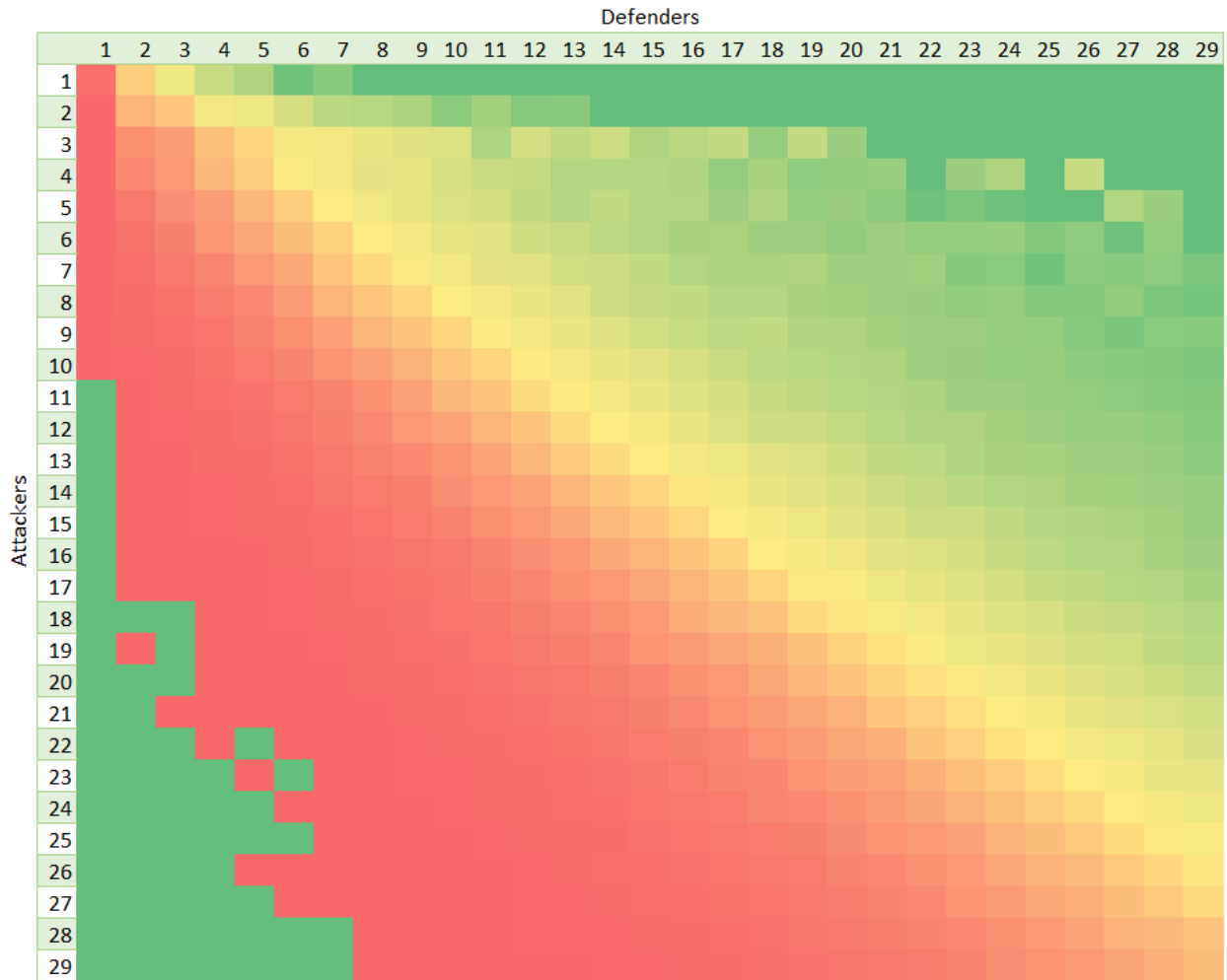


Figure 55. Heatmap of the difference between the two defensive strategies. Green indicates no change.

VITA

JACOB THOMAS GILLENWATER

Education: M.S. Computer Science, East Tennessee State University, Johnson
City, Tennessee, 2022

B.S. Computing, East Tennessee State University, Johnson City,
Tennessee, 2019

A.A.S. Computer and Information Science, Northeast State
Community College, Blountville, Tennessee, 2016

Public Schools, Sullivan County, Tennessee

Professional Experience: Substitute Teacher, ESS; Sullivan County, Tennessee, August
2021–present

Enterprise Applications Intern, Pilot Company; Knoxville,
Tennessee, May 2021 – July 2021

Graduate Assistant, East Tennessee State University, College of
Business and Technology; Johnson City, Tennessee,
August 2019 – May 2021

Undergraduate Assistant, East Tennessee State University, College
of Business and Technology; Johnson City, Tennessee,
January 2019 – May 2019

IT Intern, Uppereast Tennessee Human Development Agency;
Kingsport, Tennessee, December 2017 – August 2018

Lab Coordinator, East Tennessee State University, College of
Nursing; Johnson City, Tennessee, August 2016 – August
2018

Mathematics Tutor, Trio, Northeast State Community College;
Blountville, Tennessee, January 2015 – December 2015

Student IT Assistant, Sullivan North High School; Kingsport,
Tennessee, August 2012 – May 2014

Publications:

Jacob Gillenwater. “Combining Fitts’s Law and Angular Size to
Improve Virtual Reality User Interfaces.” Presentation,
East Coast Game Developers Conference; Raleigh, North
Carolina, April 17, 2019.

Jacob Gillenwater, Stephen Fields, Cheryl Cornett, and
Christopher Wallace. “Combining Fitts’s Law with
Angular Size.” Poster, Appalachian Student Research
Forum; Johnson City, Tennessee, April 12, 2019. 33.

Honors and Awards:

Sam Burke Memorial Teaching Award; East Tennessee State
University, graduate program, February 2021

Dean’s List; East Tennessee State University, undergraduate
program, May 2017, December 2017, May 2019.

Vice-President’s List; Northeast State Community College,
Dec. 2014, May 2015, May 2016

President’s List; Northeast State Community College, December
2015