UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

LEONARDO BOAVENTURA BOMBARDELLI

Generating Variations of the Board Game Risk Through Evolutionary Game Design

Work presented in partial fulfillment
of the requirements for the degree of
Bachelor in Computer Science

Advisor: Prof. Dr. Anderson Tavares

Porto Alegre
October 2022

“See you space cowboy.”

— Spike Spiegel

ACKNOWLEDGEMENTS

Escrevo meus agradecimentos em português para que minha família consiga lê-los sem precisar de uma tradução. Eu não teria chegado até aqui se não fosse pelo apoio constante de meus familiares. Agradeço especialmente a minha mãe, que me aturou e me apoiou nas milhares de vezes que eu surtei e achei que nunca ia conseguir pegar esse diploma na mão. Eu certamente não teria chegado tão longe sem tua ajuda.

Gostaria de agradecer também aos vários amigos que fiz no decorrer do curso. Não consigo nem descrever em palavras a sensação quase nostálgica que eu sinto ao lembrar dos perrengues que passamos juntos, virando noites fazendo trabalhos da faculdade ou estudando junto pras provas.

Por fim, agradeço também aos professores e funcionários da UFRGS, que transformaram minha passagem pelo Instituto de Informática em algo memorável. Agradeço especialmente meu orientador, Anderson Tavares, pelo apoio que me ofereceu nessa reta final do curso.

ABSTRACT

Risk is a strategy board game with multiple variations published. In this work, we propose developing a pipeline capable of producing new Risk variants using evolutionary game design. This pipeline consists of an automated playtest of the game with automated agents, the extraction of metrics from these playtests, and the usage of these metrics to generate new variations of game content through genetic programming. Through this process, we generated 80 new Risk variations. Lastly, we analyze and discuss the quality of these variations generated.

Keywords: Artificial Intelligence (AI). Board Games. Genetic Programming. Game Design.

Gerando Variações fo Jogo de Tabuleiro Risk Utilizando Design de Jogos Evolucionário

RESUMO

Risk é um jogo de tabuleiro de estratégia com múltiplas variações publicadas. Neste trabalho, nós propomos o desenvolvimento de um sistema capaz de produzir novas variações de Risk usando evolutionary game design. Esse sistema consiste em um teste de jogo automatizado do jogo com agentes automatizados, a extração de métricas destes testes de jogo e o uso dessas métricas para a geração de novas variações de conteúdo do jogo por meio de programação genética. Utilizando esse processo, nós geramos 80 novas variações de Risk. Por fim, analizamos e discutimos a qualidade destas variações geradas.

Palavras-chave: Inteligência Artificial (IA). Jogos de Tabuleiro. Programação Genética. Design de Jogos.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

AI	Artificial Intelligence
GDL	Game Description Language
GGP	General Game Player
GP	Genetic Programming
GUI	Graphical User Interface
JSON	JavaScript Object Notation
MCTS	Monte Carlo Tree Search
OSLA	One Step Look Ahead
PCG	Procedural Content Generation
RHEA	Rolling Horizon Evolutionary Algorithm
UBC1	Upper Confidence Bound 1

CONTENTS

1 INTRODUCTION

Many new board games are designed and released yearly, each applying different and innovative concepts in their design. From simple one-player games such as peg solitaire[1] to complex multiplayer games such as Monopoly[2], a good amount of experimentation and playtesting is necessary for a game designer to create successful games.

Various aspects influence how interesting a game may be for a player. A simple and easily solvable game, like tic-tac-toe, may be enjoyable for a player for some time. However, this enjoyability may quickly fade away as soon as the player notices that there is a simple combination where, if both players play it optimally, the game always ends in a draw. When this realization comes to the player, the game loses its challenging aspect for them.

Thompson (2000) analyzes that there are different attributes that a game should possess to be exciting and challenging for its players. Although some of these attributes can be of a thematic, visual, or literary nature, one of the major aspects that interest someone playing a game is the strategic aspect of it. When two players challenge each other in a chess game, each turn can be seen as a complex puzzle where both participate in discovering how to defeat their opponent. This complexity in a strategic game is not the only factor that makes it interesting, and games that are too complex can be uninteresting for most players. The elements that make games attractive to players are abstract and often unquantifiable qualities, but a vast amount of literature in game design tries to explain some of these aspects (SALEN; ZIMMERMAN, 2003).

Creating board games with the aforementioned desirable qualities is not something trivial. That is why much of game design revolves around an iterative playtesting process (FULLERTON; SWAIN; HOFFMAN, 2004), where the designer puts their game to be played by other people and can analyze how players will interact with the rules and mechanics presented to them. This process can give insight to the game designer about how players are likely to approach the game currently being developed, and it gives valuable feedback regarding how the game is in its current state.

This reliance on playtesting conducted exclusively on humans can be costly and take time. In these cases, automated agents can be employed to test the mechanics and interactions of the game. This approach is not novel to electronic games (POLITOWSKI;

[1]Peg Solitaire is a simple one-player board game, known in Brazil as *Resta Um*.

[2]Monopoly is a popular board game published by Hasbro. In Brazil, it is published by Estrela under the name *Banco Imobiliário*.

PETRILLO; GUÉHÉNEUC, 2021) and is becoming more and more common in board games. Various researchers have tested different approaches for automated playtesting in board games these last few years (SILVA et al., 2017a), and other works have proposed the creation of frameworks and pipelines for these implementations (GAINA et al., 2020). Some of these works also revolve around creating game description languages (GDL), such as Genesereth, Love and Pell (2005). These are description languages capable of expressing rules and mechanics for a plethora of different games. Alongside GDLs are general game players (GGP), software systems capable of performing well across different games.

Browne and Maire (2010) propose an innovative approach to game design called evolutionary game design. In it, the authors created a framework for the automatic creation of novel board games. This can be accomplished through the Ludi framework, created by the authors to manage an automated playtesting of games defined by a GDL and the gathering of metrics that quantify the several attributes that make a board game enjoyable to players. These metrics are then used as input in a evolutionary algorithm that generates new games by modifying the rules description of the original game. This work created Yavalath[3], which was published as an original and novel game. The games generated through the Ludi framework are restricted to the games describable by the Ludi GDL, which limits the games generated in this work to only combinatorial board games.

In the present work, we aim to validate the usage of Browne's concept of evolutionary game design to generate new variations of modern non-combinatorial board games. We propose to apply evolutionary game design to create new variations of a non-combinatorial board game. We apply these concepts to generate new variations of a simplified version of the game Risk[4]. To accomplish this, we present a pipeline that can automatically playtest the game and capture the measurements of different quality criteria for these playtests. We can then use these measurements as input to a evolutionary algorithm capable of generating new variations of the original game.

Our work's approach shows many benefits compared to Browne and Maire (2010), such as:

- Specializing ourselves to only one type of game, instead of a variety of simple board games used by Browne and Maire (2010), allows us to represent more complex

[3]Yavalath is a board game created through the Ludi framework and published by Browne. More of it can be seen at https://boardgamegeek.com/boardgame/33767/yavalath.

[4]Risk is a multiplayer board game published in the United States by Hasbro. In Brazil, the game is published under the name *War* by Grow.

games in this pipeline.

- Using this approach, game designers can not only create novel variations of a board game but also design new types of board games with innate randomness of content in their design, similar to the usage of procedural content generation (PCG) in digital games (SHAKER; TOGELIUS; NELSON, 2016).
- Separating the base rules of the game, which are static, and other parameters like the game's map allows us to create a procedurally-generated board game that can generate different content for its players based on qualities like game balancing or length of playtime.

The remainder of this work is as follows. Chapter 2 provides information about key concepts necessary for a better understanding of the proposed pipeline, as well as the discussion of works that approach similar problems. In Chapter 3, we present the pipeline for the generation of board game variants, our proposed solution for using evolutionary game design to create new content for modern board games. In Chapter 4, we present the results of our implementation. At last, Chapter 5 presents the final considerations regarding this work.

2 BACKGROUND AND RELATED WORKS

This chapter introduces fundamental concepts regarding board games, game elements representation, automated playtesting, and evolutionary game design, all necessary to understand the present work. While introducing these concepts, we also overview and discuss related work, their contributions, and their limitations.

2.1 Games

There are many different definitions of game. Of these definitions, we use the one proposed by Salen and Zimmerman (2003), p. 80:

> A game is a system in which players engage in an artificial conflict, defined by rules, that results in a quantifiable outcome.

Separating each key element of this definition, we have:

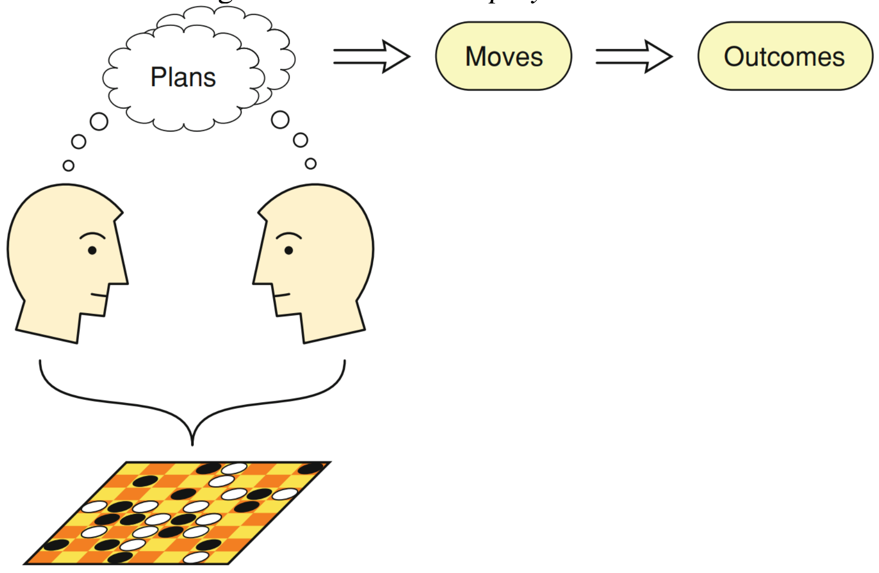
- *Players*: A game is something that one or more agents actively take decisions to play. Players are the agent that interact with the system of a game.
- *Conflict*: All games must embody some contest of powers. From cooperative to competitive games, from solo to multiplayer games, every game must provide some kind of conflict for the player(s) to solve.
- *Rules*: Rules play a crucial part of games, since they provide the structure out of which players can interact with the system, delimiting what they can and cannot do.
- *Quantifiable Outcome*: Games must have a quantifiable goal and outcome. Each game must have a goal, a state in which a player has either won, lost, or received some kind of score that reflects their actions.

When analyzing the aforementioned definition from Salen and Zimmerman (2003), Browne (2011) creates a simplified model of what a game is, called *means-play-ends* model. A visual representation of this model is presented in Figure 2.1, and it consists of:

- *Means*: The equipment and rules used to play the game.
- *Play*: The interaction between players, which is defined implicitly by their plans and explicitly by the moves made by them.
- *Ends*: The outcome that each move produces in the game.

The definitions and abstractions presented above are useful for us in a couple

Figure 2.1: The *means-play-ends* model



Source: Browne (2011)

of ways. First, they allow us to separate the games under this work's study from other types of literary or interpretative games (like roleplaying games, for example). Second, they define games as systems with specific elements that will be important in this work. Elements such as having a specific goal in which the players compete or cooperate to achieve quantifiable goals are important when we design the playtest of a game.

Of all types of games that can be represented in the definitions mentioned above, a substantial amount of works in the literature focus on a specific subset called *combinatorial games*. These games are:

- *Finite*: Outcomes are well-defined and the game ends in a finite number of turns.
- *Discrete*: They are turn-based, where each player acts in a different turn.
- *Deterministic*: There is no randomness or chance in the game.
- *Perfect Information*: The whole game state is always visible for both players.
- *Two-players*: In games with more than two players, coalitions between players may arise, which can bring a social aspect to the game, which is out of the scope of combinatorial games.

Combinatorial games are interesting because they provide a playground for a multitude of mathematical and computational analysis for the creation of agents capable of beating their opponent on them (FRAENKEL, 2004). Although our work is not a work on combinatorial game theory, many related works cited here are focused on the study of these types of games.

Another important distinction that has to be made is the concept of *game distance*. We use the definition provided by Browne (2011), p. 9, to determine what is a game variant or an entirely new game:

> The distinction between a variant and a new game is subtle but may be achieved by representing both games as rule trees and performing standard tree comparison to find the difference between them. Differences between rules would be weighted more heavily, while differences between their attributes (would be) weighted more lightly.

In this definition, changing a game like Tic-Tac-Toe from a board with three tiles to a board with four tiles would be a simple variant of the game. On the contrary, changing the board tiling from a square to a hex would cause us to modify the condition of when the game ends, resulting in an entirely new game.

2.2 Game Elements and Representation

A vital step in proposing a pipeline capable of testing and generating new content for a game is how a game and its elements are represented. To do that, we need to define abstractions for each of the game's components and how they interact with each other through the game's rules. Next, we analyze how different works approach this problem.

Genesereth, Love and Pell (2005) created a logic programming language to provide an environment for evaluating general game players (GGP) in a diverse array of combinatorial games. This game description language (GDL) is the conceptual basis for many other works in the area, including Ludi (BROWNE, 2011). This logic programming language is similar to Prolog, purely declaratory, and offers restrictions that assure that all questions of logical entailment are decidable.

Browne (2011) uses the concept of *ludemes* as a building block for the definition of games. A *ludeme* is a fundamental element of play, often representative of a rule, equivalent to a game's component. *Ludemes* are notable because they are elements of a game design that, according to the author, usually pass from one game or game class to another. An example of two *ludemes* in the Ludi GDL can be seen in Figure 2.2. The first *ludeme* declares that the tiling of the board will be composed of square tiles, while the second declares that the size of this board will be 3x3.

The concept of a *ludeme* is interesting because it allows us to relate different *ludemes* with another one, generating higher levels of a compound *ludeme*. That can be seen in the Figure 2.3, where the definition of a 3x3 board with square tiling is presented:

Figure 2.2: *Ludeme* in the Ludi GDL

```
(tiling square)
(size 3 3)
```

Source: Browne (2011)

Figure 2.3: *Ludemes* in the Ludi GDL composing a higher level *ludeme*

```
(board
  (tiling square)
  (size 3 3)

)
```

Source: Browne (2011)

The abstraction of a *ludeme* is essential for the work of Browne because it allows the succinct description of games. This view of game rules as interchangeable and encapsulated *ludemes* allows the author to propose ways to create new games using these rules via genetic programming. An example of Tic-Tac-Toe in the Ludi GDL can be seen in Figure 2.4.

Figure 2.4: Tic-Tac-Toe in the Ludi GDL

```
(game Tic-Tac-Toe
    (board
       (tiling square)
       (size 3 3)
    )
    (win (in-a-row 3))
)
```

Source: Browne (2011)

As shown in Figure 2.4, the Ludi GDL infers considerable information from the description of the games. The description of Tic-Tac-Toe has no *ludeme* related to how players interact with the board, so the GDL infers that, if no specification is provided, the players will take turns placing one piece on empty tiles. Although this is useful for the easy specification of simple games, it cannot be used to describe complex games.

Gaina et al. (2020) presents the Tabletop Games framework (TAG), a Java-based platform for the benchmark of GGP in modern board games. To do that, TAG provides the user with a common template for implementing board games, standardizing agents' access to the games on the platform. The board games in the framework, instead of being described in a declaratory language like in the works discussed above, are written in Java classes that inherit base classes of the tool. The most important classes are the following:

- *Game State*: This class contains all the necessary information to describe the game's state. These can range from cards in the hands of players to the number of points that a player has and is used by the agent to choose their actions.
- *Forward Model*: This class is responsible for setting up and advancing the game state when provided a player action and computing all possible actions a player can perform in the current game state. In this sense, since it is responsible for modifying the game state, it indirectly implements the rules of the game.
- *Actions*: This class represents every decision the agents must make in a game, from moving pieces to drawing cards.

TAG's framework introduces many relevant aspects regarding the representation of games to the present work, such as:

- The capability to represent complex multiplayer games, cooperative games, and games with hidden information.
- A simple structure that allows the creation of a graphical user interface (GUI) for the board games.
- The separation of the game's rules and parameters. Each of the games currently implemented in the framework has a list of valid parameters in which they can be instantiated. Tic-Tac-Toe, for example, can be instantiated in TAG with a 3x3 grid, a 4x4 grid, or a 5x5 grid. This separation makes it easier to implement variations of a game on the platform.

2.3 Automated Playtesting

The TAG framework (GAINA et al., 2020) implements four different agents in the platform. All games implemented in the platform require an heuristic that, given a game state, returns a numerical value that indicates how close a player is to win. The agents then use these heuristics to decide their actions. Two of the simplest agents implemented are the random agent, which simply chooses actions at random, and the One Step Look Ahead (OSLA) agent, which evaluates all possible actions from a single game state and picks the one which leads to the highest valued state. The other two agents implemented are Rolling Horizon Evolutionary Algorithm (RHEA) (PEREZ et al., 2013) and Monte Carlo Tree Search (MCTS) (BROWNE et al., 2012). Since the purpose of the platform is to measure the performance of GGP in modern board games, it allows for tournaments between agents.

Mugrai et al. (2019) proposes the implementation of procedural personas for the playtesting of games. These personas are created by modifying MCTS where, rather than applying the Upper Confidence Bound 1 (UCB1) formula in the exploration of moves, the authors use genetic programming to evolve a persona-specific formula (HOLMGÅRD et al., 2018). Using this variation of MCTS, the authors tested these agents in a Match-3 game (similar to Candy Crush[1]). They developed four different personas, each following a different type of playstyle: one that tries to maximize the game score, one that tries to minimize the game score, one that tries to maximize the available number of moves, and one that tries to minimize the available number of moves. The work also conducted a user study with human players to establish a baseline. In the results, the authors analyze that the performance of these different agents is a good way to define an upper bound and a lower bound of the performance of an average player.

Silva et al. (2017b) presents an AI-based playtesting for the game Ticket to Ride[2]. Since the game has a massive amount of states and is an imperfect information game, agents using techniques such as A* or MCTS present a poor performance. Because of that, the authors created four different rule-based agents with different playstyles. They experimented with these agents on all 11 different commercially released variants of the game and could analyze which different strategies would be dominant in each game variation. The work also analyzes specific aspects of the game's original map, like the most common routes and cities owned by the player who won the game. The authors also managed to find, in the playtest, scenarios not covered by the rules of the game.

[1]Candy Crush is a popular mobile game where the user has to move pieces in order to match three or more pieces of the same type.

[2]Ticket to Ride is a multiplayer board game published by Days of Wonder. In it, players must create railways between cities in a map.

2.4 Evolutionary Game Design

Browne (2011) proposes a pipeline for the generation of novel games through automated playtest, evaluation, and synthesis of new games. The framework Ludi is structured by the following components:

- *GDL*: defines the scope, structure and rules of the game.
- *GGP module*: interprets the rules of a given game and coordinates which actions to take.
- *Strategy module*: informs move planning.
- *Criticism module*: measures the game quality.
- *Synthesis module*: generates new games through genetic programming (GP) (KOZA, 1992).

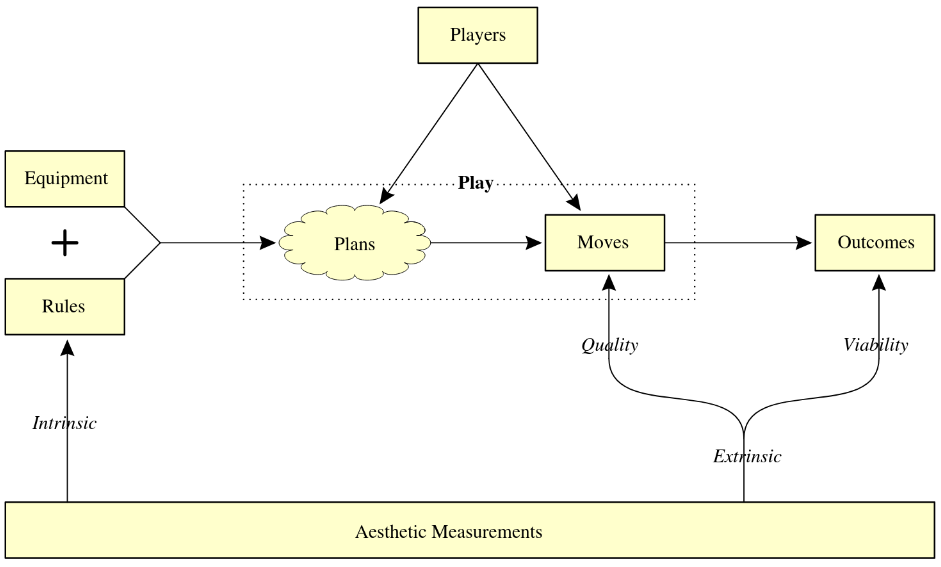
Since the present work focuses on creating content for a specific game, we will skip Ludi's general-purpose game generation aspects, primarily represented by the GGP, strategy, and synthesis modules of the pipeline. These modules are focused on playtesting generic games with different rules, and since we only deal with a game with constant rules, they do not fit our work. The Ludi GDL has already been explained in Section 2.2, so here we discuss the game evaluation and how they are used in the synthesis of new games.

The criticism module of Ludi measures games based on certain aesthetic criteria. Browne (2011) primarily uses the definition of Thompson (2000) of which attributes a game should be interesting to players. Brownie separates these aesthetic criteria into three categories, shown in Figure 2.5, which are:

- *Intrinsic*: based on rules and equipment.
- *Viability*: based on game outcomes.
- *Quality*: based on trends of play.

The game's rules define intrinsic criteria, which can be the win condition of a game of the tiling type of a board, for example. These criteria, though, are not the focus of discussion in our work since the game's rules will be static. Viability criteria are those based on the game's outcomes. These are responsible for checking if a game is balanced or the average turn duration of the game. Last, the quality criteria attempt to measure the players' engagement with the game through their moves' effects on the game. These are generally more difficult to measure and require a lead history of the playtest to be

Figure 2.5: Ludi aesthetic model of games



Source: Browne (2011)

calculated. Lead history estimates which player is ahead in the game, generally with an heuristic evaluation. In Browne's work, the authors define 57 aesthetic criteria: 16 intrinsic criteria, 30 quality criteria, and 11 viability criteria.

An essential aspect of Browne's work is the game ranking experiment done with human players, where players rated games after playing against an AI. Authors ranked the games according to this data and used the rankings do derive the 57 aesthetic quality criteria. The criteria that best correlated with the game rankings were then used to calculate the fitness function of the games in the evolutionary programming process.

2.5 Comparison of Related Works

This section will briefly discuss each related work analyzed here and compare their proposals to ours. Following the structure of Section 2, we will separate this analysis into the three elements discussed here: game elements and representation, automated playtesting, and evolutionary game design.

Section 2.2 analyzed how different works represent different board games. Genesereth, Love and Pell (2005) proposes a declarative language used to declare only combinatorial games for testing GGP, but it is an important work that inspired the creation of the Ludi GDL. Browne (2011) represents games using a declaratory language, but the elements used to construct games are interchangeable to generate new games. This language enables the Ludi framework to represent combinatorial games with a small amount of information but is limited in its capacity to represent more extensive and complex games. Gaina et al. (2020) uses Java code to codify the elements of the games represented but offers abstractions such as game parameters that allow us to modify certain aspects of them without interacting with the code. Our work represents only a single game and all possible variations by implementing something similar to Gaina et al. (2020), but our focus is on the game variants generation, not on playtesting.

Section 2.3 analyzed the approach of three different works related to automated playtesting. In Gaina et al. (2020), the TAG framework implements four different GGP used to playtest their games. Mugrai et al. (2019) implemented four procedural personas for a single game's playtest. These personas were used to evaluate players' lower- and upper-bound performance in the game. Silva et al. (2017a) created four rule-based AIs to extract metrics regarding the optimal strategy of different game variations. In our work, we use one rule-based agent to extract the metrics that will then be used to generate new games in the evolutionary game design pipeline.

Section 2.4 analyzed the concept of Browne (2011) of evolutionary game design. In it, the author generates new simple combinatorial games through the Ludi framework. As mentioned before, our works differ in that this work focuses on implementing these concepts in generating variations of a single non-combinatorial modern board game.

3 PIPELINE FOR GENERATING RISK VARIANTS

This chapter presents our pipeline for the generation of Risk variants. We begin by defining the rules and specific aspects of the board game we will implement (Section 3.1). We then explain the proposed pipeline in broader terms (Section 3.2). After that, we focus on three important aspects of the pipeline: the automated playtest (Section 3.2.1), the aesthetic measurements we define to measure quality in board games (Section 3.2.2), and the usage of genetic programming to generate new board game variations (Section 3.2.3).

3.1 Simplified Risk

In order to validate the idea of evolutionary game design for the creation of more complex games, we use a simplified version of the game Risk. Risk is a board game published by Hasbro that has many variations, both in terms of rules and game stylization. Risk is a territorial control game where players control armies to conquer enemy territories and take control of continents, trying to accomplish specific conditions given at random from cards. Players take turns attacking other continents with their troops until a player controls all territories of the map, which is then declared the winner. Due to its overall popularity and variety of published content, the game has been chosen for this work. Figure 3.1 shows the accessories and gameboard in a physical copy of the original game.

Figure 3.1: Risk Board Game

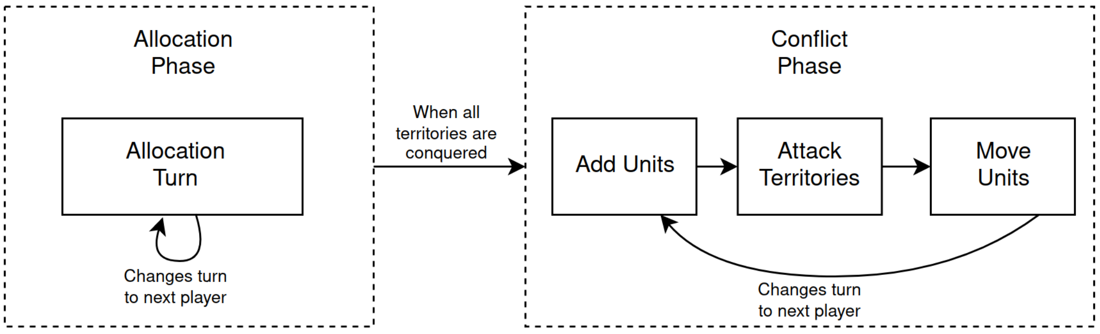
Source: BoardGameGeek

Our game variation aims to reduce the number of possible states and simplify the execution of the playtest agents. To accomplish this, we removed the hidden information aspects of the original game, which were cards that players would use and exchange for troops in random territories. We also changed the game's combat mechanics, giving an advantage to the attacking player and encouraging aggressive behavior from players.

In our version of Risk, a player wins by conquering all territories across a map. Each territory belongs to a continent, and players must mobilize troops across these territories to invade enemy territories. Players roll dices to decide which units win the conflict. At the beginning of their turns, players will recruit new units to put in territories of their choice, and bonus units will be recruited according to the number of continents a player controls.

A game in this simplified version of Risk is comprised of two game phases: an *allocation phase*, where players will take turns choosing their starting territories, and a *conflict phase*, where players will take turns recruiting new units, capturing enemy territories, and moving units at the end of their turns. Figure 3.2 shows a simple schematic of the game execution.

Figure 3.2: Simplified Risk Game Loop


Allocation Phase
Allocation Turn
Changes turn to next player
When all territories are conquered
Conflict Phase
Add Units
Attack Territories
Move Units
Changes turn to next player

Source: The Author

Since most of the game actions revolve around the conflict phase, we will use the definition of a "turn" to define a whole turn in the conflict phase. In that sense, we can think of the allocation phase as a setup for the beginning of the game. A turn in the conflict phase has three stages:

- *Add Units*: At the beginning of each turn, the player will add new units to their territories. This number is based on the number of territories they control plus the bonus troops they get from controlling a whole continent. All these troops can, then, be added to any of the player's controlling territories of their choosing. An important exception is that, in the first turn of the game, the player only adds troops based on their territories, not on their continents. This is done to minimize the first-move advantage in the game.

- *Attack Territories*: After adding new units, the player can attack enemy territories. When a player has more than one unit in a territory adjacent to an enemy territory, they can attack it. The attacking player rolls dice up to the number of attacking units, at a maximum of three units, and orders the values of the dice rolled from higher to lower. The defending player rolls dice up to the number of defending units, at a maximum of three, and orders these dice as well. Then, the players compare each value of these sorted arrays of dice, and the player who rolled lower loses a unit. In the case of draws, the player who defended loses a unit, which gives the attacking player an advantage in these combats. If the defending player loses all units in a territory, the attacking player conquers the territory. When conquering a new territory, the attacking player must send one unit from the territory where the attack comes to the newly conquered territory.
- *Move Units*: After the combats in the previous step, the player may move units from territories adjacent to each other. The player may only move units from territories they own to territories they own, and they must always leave one unit at each owned territory.

After these steps, the player passes their turn to the next player. This game loop repeats until a player owns all territories on the map.

In earlier steps of the pipeline development, we replicated the favoring of defensive troops in combat draws. The length of these games, when tested with our agents, would sometimes get into a stalemate of hundreds of turns. Because of that, another modification in the original game was altering the draw in combats, which now favors the attackers instead of the defenders. Making this modification means rewarding aggressive players and discouraging defensive players, making stalemates rare.

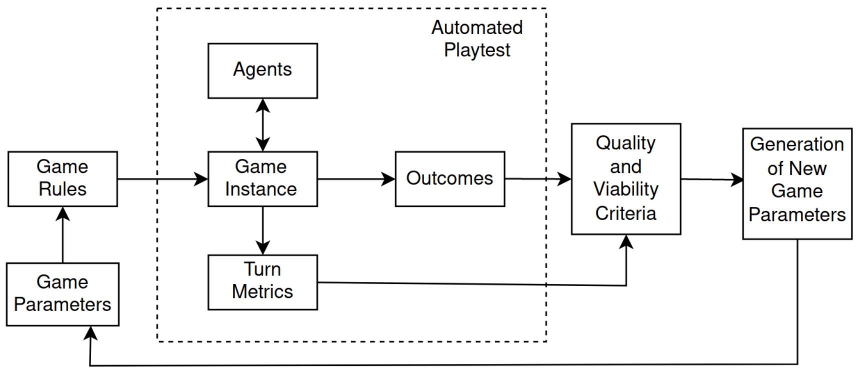
3.2 Proposed Pipeline

To validate the idea of applying evolutionary game design to generate variations of a board game, we implemented a pipeline in Python available on GitHub[1]. The pipeline implemented to generate new Risk variations can be broken down into three major steps:

[1]https://github.com/LeonardoBombardelli/Risk-Content-Generation

the automated playtesting of the game, the gathering of metrics in order to check the quality and viability of the game tested, and the usage of these metrics to generate new variations of this game. A simplified visualization of this proposed pipeline can be seen in Figure 3.3. The remainder of this subsection will give an overview of the current implementation of this pipeline.

Figure 3.3: Overview of Proposed Pipeline

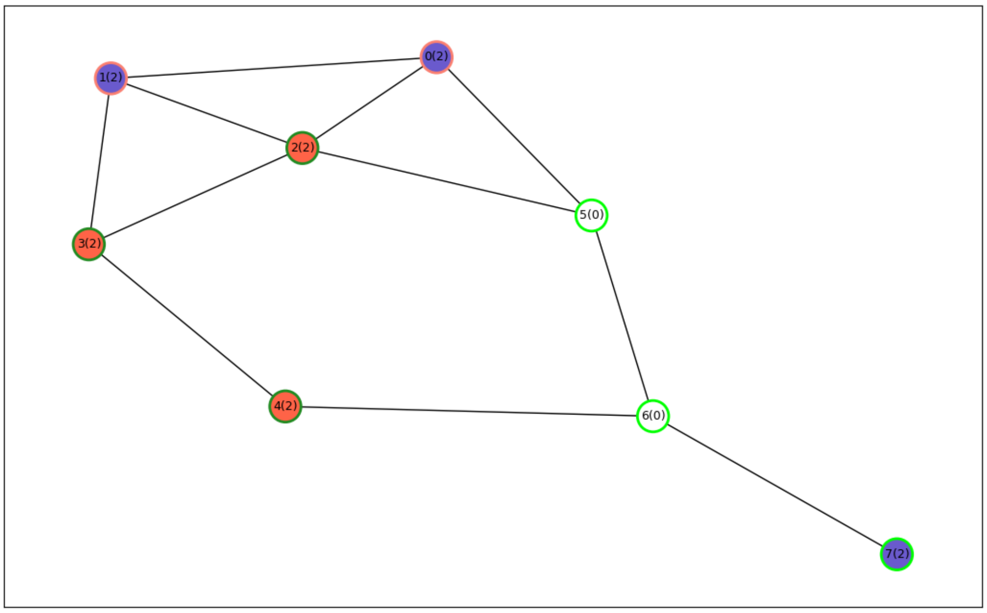


Source: The Author

We have a modularized implementation of the simplified Risk, separating the game rules and parameters. The parameters are: the game map, represented as a undirected graph where each node represents a territory, the continents, represented as a list of territories that compose each continent, and how many units the ownership of the whole continent provides to the player, represented as an integer. These parameters are modified in the pipeline to generate new game variations. These parameters, represented in a JSON file, are loaded into the game, where the user can see a visual representation of the map and its elements.

Figure 3.4 shows a small map of the simplified Risk in our implemented user interface. Each node of the map represents a territory, where inside it are two numbers: the left one is the node identifier and the right one, under parenthesis, is the number of troops currently present in the territory. The color on the outline of each node represents to which continent it belongs. When a player captures a territory, the color inside the node changes to the color of the player who captured it.

Figure 3.4: Implemented Map GUI



Source: The Author

3.2.1 Automated Playtest

After loading the game parameters, we have an instantiated game that can be playtested. The playtesting of this combination of game rules and parameters can be repeated multiple times to obtain more precise metrics for the quality and viability evaluation. In this playtesting, the following cycle repeats until the end of the game:

1. The game returns a game state to the agent responsible for taking actions for the current player. Then, the player performs their actions until the end of their turn.
2. Using an heuristic, the game evaluates the current game state to gather metrics for the calculation of the quality criteria and records this data for future usage.
3. The game changes the current player.

When a player wins the game, we log the winner's data and turn count to use for the calculation of the viability criteria later. All the records of the playtest are then passed along to the Quality and Viability Criteria step of the pipeline.

There are two essential elements in the playtesting process that are important to consider: the heuristic used to determine how ahead a player is from another and the agent used to make decisions in the game. These elements are modularized in this implementation, meaning they can be modified without changing the rest of the pipeline's structure.

We currently use Equation 3.1 as the heuristic, where $h(i)$ is the heuristic value for player i, $terr_i$ and $terr_{total}$ are the territories of player i and the total number of territories, respectively, and $unit_i$ and $unit_{total}$ are the number of units of player i and the total number of units, respectively:

$$h(i) = \frac{terr_i}{terr_{total}} * w_{terr} + \frac{unit_i}{unit_{total}} * w_{unit} \tag{3.1}$$

w_{terr} and w_{unit} are values that serve as weights to each components of the equation. Although this heuristic is a simple weighted proportion of the player's current territories and units in relation to the total amount, it is enough to measure how ahead a player is compared to another.

At the start of the development of the pipeline, we tried to create an agent using Vanilla MCTS (BROWNE et al., 2012). This agent, though, presented many problems that made us realize the difficulties of creating a competent agent capable of playing the simplified Risk. The poor performance of the agents when using MCTS revolved around the massive amount of game states the agent would need to process to arrive at terminal states. Even using a small version of the map developed by us, with eight territories in total, and letting 30 seconds for the agents in each decision, they still had a degenerated defensive behavior where they would only recruit new units and skip their turns. Given that, in each turn, the agent has to make various small decisions, this approach would not be feasible in the playtesting.

Difficulties in the creation of capable agents for board games were reported in the work of Blomqvist (2020), where the author proposes a Risk agent developed using a variation of AlphaZero. However, the author points out that the agent's performance is subpar compared to human players. In the work of Silva et al. (2017b), the authors circumvent the problem of a poor performance by GGP agents by using a rule-based agent capable of playing the game instead. Based on their work, we developed a simple rule-based agent as well.

In the allocation phase, our agent attempts to capture the continent that yields the highest number of troops. If no entire continent is available for the agent, it chooses a random empty territory in his neighborhood. If no adjacent territory is empty, it chooses a random non-adjacent territory.

In the conflict phase, the agent has a greedy and aggressive playstyle where:

- In the addition of new units, they distribute all new units to territories adjacent to enemy territories where troop values are lower or equal than three, or the lowest value of them, whichever is lowest.

- When attacking, it always attack neighboring territories with fewer units, or if their own territory has three or more units. Ties are broken by selecting the lesser defended territory.
- When moving troops, they always move troops to the territories with fewer units adjacent to enemy territories, leaving only one unit in territories that have no frontier with enemy territories.

3.2.2 Aesthetic Measurements

As explained in Section 2.4, we use Browne's definition of aesthetic criteria to evaluate the quality of the game variants generated. These criteria are separated into two categories based on what they measure: viability criteria, which measure attributes based on the game's outcomes, and quality criteria, which measure attributes based on the player's engagement with the player. We run the automated playtesting multiple times per game variation to gather enough data to make these calculations. From each instance of playtesting, we gather the data of both players' heuristics values from the game state of each turn, the game's winner, and the number of turns. They are then used to measure four criteria, two regarding game viability and two regarding game quality, which will be described in this subsection.

3.2.2.1 Viability Criteria

Advantage (C_{adv}) is a measurement given by the ratio of wins for the first player to move compared to the second player. It is used to measure if the game favors a specific player in the turn order and is given by Equation 3.2, where $wins_f$ is the number of wins the first player to move has and $wins_s$ is the number of wins the second player to move has. A game with an C_{adv} value of 0.5 is a balanced game with no advantages. Lower values describe a game that favors the first player to move, and higher values the second player.

$$C_{adv} = \frac{wins_s}{wins_f + wins_s} \tag{3.2}$$

Duration (C_{dur}) is a measurement representing the difference between the number of turns a game has and the desirable number of game turns. It is defined by the average absolute deviation of each game length T_g from the preferred game length T_{pref} across all games playtested G. Equation 3.3 presents this criteria. A game with a lower C_{dur} means a game with its length closer to the preferred length. In contrast, a higher value means a game with lengths more distant from the preferred length.

$$C_{dur} = \sum_{g=1}^{G} \frac{|T_{pref} - T_g|}{T_{pref}} \Big/ G \tag{3.3}$$

3.2.2.2 Quality Criteria

Drama (C_{dra}) is a measurement representing the amount of disadvantage a player that ultimately won a game had. This metric indicates how possible is for a player to recover from bad positions in the game. Drama is calculated by taking only the measurements of the turns where the winning player had a disadvantage. We then calculate the difference between both players at those moments. This criteria is represented in Equation 3.4, where $h_w(t_n)$ and $h_l(t_n)$ are the heuristic values for the players who won and lost the game, respectively, at a given turn t_n. $count$ is an operation that returns the number of turns where this condition is true. This operation is done across all G games where the playtest was realized. For drama, higher values mean a higher chance of the player being able to come back from bad positions in the game, whereas lower values mean the opposite.

$$C_{dra} = \sum_{g=1}^{G} \frac{h_w(t_n) < h_l(t_n)\{\sqrt{h_l(t_n) - h_w(t_n)}}{count(h_w(t_n) < h_l(t_n))} \Big/ G \tag{3.4}$$

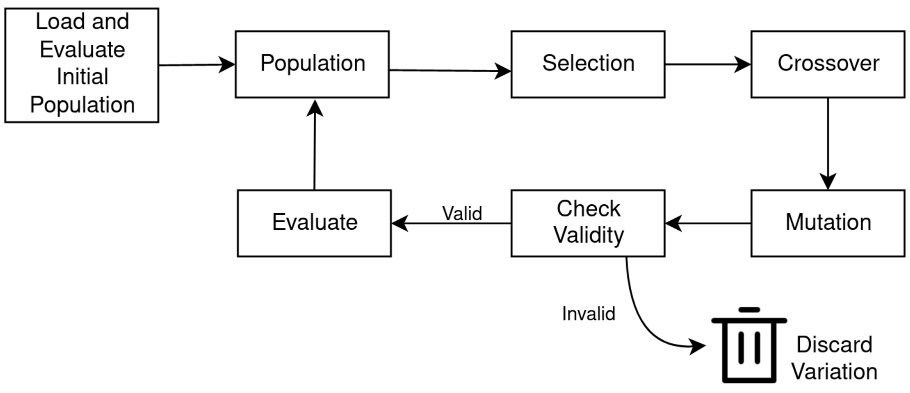
Lead Change (C_{lc}) is a measurement that shows how much the lead of the game changes between players. In it, we count how many times the game lead changes between players and divide it by the number of turns the game had. The equation for this criterion is expressed in Equation 3.5, where the $leader$ operation returns the player whose heuristic had a higher value in a given turn t_n. $count$ is an operation that, same as before, returns the number of turns where this condition is true. G is the total amount of games simulated in the playtest, and T is the number of turns that game had. With this metric, higher values mean more lead changes occur in the game, whereas lower values mean the opposite.

$$C_{lc} = \sum_{g=1}^{G} \frac{count(leader(t_n) \neq leader(t_{n-1}))}{T} \Big/ G \tag{3.5}$$

3.2.3 Genetic Programming

Genetic Programming (GP) is a group of methods that borrow concepts from natural evolution to produce computer programs representing particular solutions in the overall solution space (KOZA, 1992). In this work, we use it to generate new variations of the simplified Risk. This process is shown in Figure 3.5 and explained below. It follows the same structure and general concepts as those implemented in Browne (2011) but is adapted to the generation of variations of the simplified Risk.

Figure 3.5: Game variation generation overview

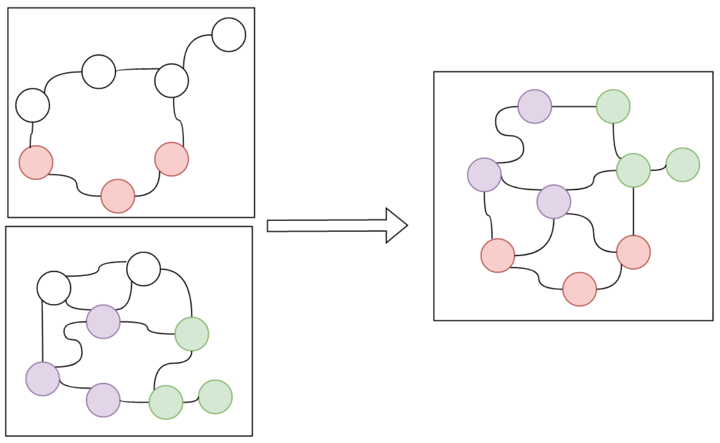


Source: The Author

Our initial population is composed of six different map variations created by us based on the game's original map. We sort the population by their fitness value, described below, and select two of them as parents through stochastic universal sampling (BAKER, 1987). Using this, we have a chance to select individuals from the entire population, but those with a better fitness measurement will be selected more often. Our fitness function determines only the sampling process, so no individuals are removed from the population. We keep individuals with low fitness to encourage diversity in the generation of new individuals.

After selecting two individuals from the population, we use our crossover function to generate a new individual inheriting characteristics from both. This crossover is done by selecting some continents from the first and second parents and combining them in a new map, creating new connections between these continents' territories. Now on the new map, these continents will inherit the same territory structure and bonus units that the parent's continent has. Figure 3.6 illustrates the crossover operation between two maps, with the coloring representing the continents being chosen and propagated to the new map.

Figure 3.6: Crossover operation between two maps



Source: The Author

After a new candidate has been generated through the crossover operation, we mutate it. We have defined five operations of mutation, each having a 20% chance of occurring in an individual, which are:

- Creating a new connection between two territories;
- Removing an existing connection between two territories;
- Moving a territory from one continent to another;
- Swapping the value of provided units of two continents;
- Modifying the value provided units of a continent by one.

In the crossover and mutation process, an individual may be modified to the point that it cannot be a valid variation to the game. Because of that, we need to check if the generated individual is a valid parameter by checking two characteristics: if the graph generated by the territories' connections is planar and connected. The reason is that territories' connections represent the map on which the game is played, so it must be a planar graph. Also, the players must be able to conquer all territories to win the game, so all territories must be reachable. If an individual fails in either of these tests, it is discarded, and we revert to the selection of new parents.

After generating a valid individual, we evaluate it through our automated playtest. In it, we run the game with our agents a hundred times, gathering the turn metrics and outcomes of each game. After this playtest, we calculate the individual's quality and viability criteria and compare them with the estimated optimal criteria values (discussed in Chapter 4). This comparison of the individual's and optimal criteria is calculated through the Euclidean distance given by the Equation 3.6, where M is the criteria calculations measured, I is the ideal criteria values, and C is the number of different criteria. We use this evaluation to define the distance between the new individual and an ideal individual. Since our fitness function is inversely proportional to this distance calculated, we normalize this value across all other individuals of the population and calculate our fitness function as $1 - d(M, I)_{normalized}$.

$$d(M, I) = \sqrt{\sum_{c=1}^{C} (M_c - I_c)^2} \tag{3.6}$$

4 EVALUATION

In this chapter, we present and analyze the results of our proposed pipeline for the generation of Risk variants. We begin by analyzing the quality and viability criteria of the simplified Risk with our base game parameters (Section 4.1). We then choose the preferred values for these criteria and use them to generate new game variants (Section 4.2). Still in this section, we analyze the game variants generated, discussing the individuals with the best and the worst performance. Lastly, we discuss the obtained results (Section 4.3).

4.1 Setting Values for Optimization

For our proposed pipeline to generate new game variations, we need to provide it the preferred values for each of the calculated criteria. These values are necessary for the fitness calculation of each game variation generated and impact which kind of content the pipeline will generate. Browne (2011), when proposing the framework for evolutionary game design, realized an experiment with human players that would later be used to define the criteria and values used in Ludi's fitness calculation. This experiment is explained in Section 2.4. Our approach is more simplistic, given that the time restrictions and scope of this work made it impossible to playtest the simplified Risk with players.

Another set of values we need to set before experimentation are the weights for the heuristic calculation from Equation 3.1. We use w_{terr} and w_{unit} as 0.8 and 0.2, respectively. Since the control of more territories puts the player closer to the victory condition and offers them more units in the next turn, they are given a higher weight on the formula.

In this work's experimentation, we ran 500 playtests using the game's original map and parameters, with 42 territories and 6 continents. This experiment took around two minutes, running in a single-threaded Python application. After these playtests, we gathered their data and calculated the game's current advantage, drama, and lead change criteria, as well as the average game duration. These metrics, which will be analyzed below, are used as an insight to determine the ideal criteria values for generating new game variations.

Table 4.1 shows the measurements gathered from the playtests run in the original Risk map. We have the four metrics gathered, their average value, and their standard deviation (when applicable). From these metrics, we can notice that the parameters of the original Risk, when applied to our simplified Risk, is an unbalanced game. We can see this lack of balance explicitly in the Advantage (C_{adv}) criterion. Advantage gives us the information of how probable the first player to move in the game is to win it—an Advantage of 0.5 means that the game is perfectly balanced. A value of 0.252 in this criterion means that, in all 500 playtested games, the first player to move won 74.8% of them.

Table 4.1: Metrics from the original Risk map

Metric Measured	*Average Value*	*Standard Deviation*
Advantage (C_{adv})	0.252	Not Applicable
Drama (C_{dra})	0.064	0.108
Lead Change (C_{lc})	0.074	0.148
Average Duration	15.558	4.681

Source: The Author

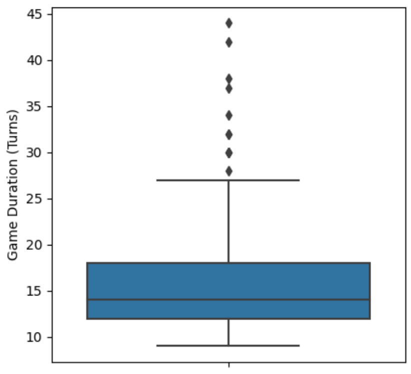
A possible explanation for the first player's noticeable advantage can be seen in the Drama (C_{dra}) and Lead Change (C_{lc}) values in Table 4.1. Lead Change shows us how frequent the change in the lead in the game is—values closer to zero mean that players rarely change leads in the game. Drama values indicate how severe the lead disadvantage a player who ultimately won the game had. Given that both values are closer to zero, we can assume that players who get a lead in the game rarely lose it, and when they do, they rarely manage to get it back. The standard deviation of these metrics is high because many games had Lead Change and Drama values equal to zero (meaning the first player always had the lead), corroborating this explanation. Our hypothesis for this unbalance is that the first player in the game always has a chance to capture enemy territories and get ahead, leading to a *snowball effect*[1].

Analyzing the Average Duration of the games in Table 4.1, we can see that games had an average duration of around 15 turns. However, the high standard deviation of the metric indicates that some games can have a significant fluctuation in their game duration. Figure 4.1 shows a boxplot representing the game duration of each playtest where we can analyze this behavior. We can see in the figure that more than half of the games end before 15 turns, and this distribution becomes more and more sparse as the number of turns increases. Outliers in the graph show us that we still have games going far beyond their average turn length, even with modifications in the game's rules implemented to prevent stalemates.

In Figure 4.2, we can analyze the turn duration of the games separated by the

[1]In games, snowball effect is a term used to describe when a player uses a slight initial advantage to gain more advantage throughout the game.

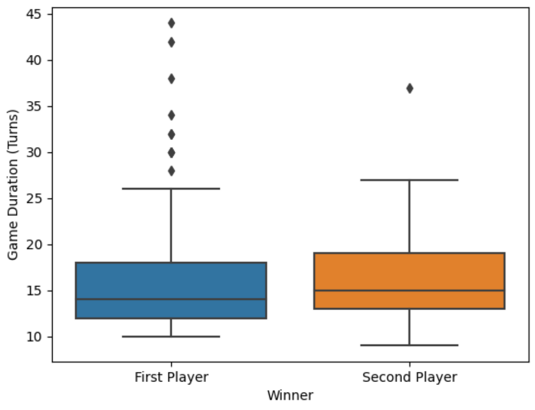
Figure 4.1: Boxplot of game duration (in turns) in the original Risk map



Source: The Author

game's winner. In it, we can see that the median number of turns from the games where the second player won is two turns higher than those where the first player won. The snowball effect corroborates this correlation between game duration and higher turn counts. This is because games where the first player could not dominate the map in the first few turns and snowball their advantage take longer to complete, giving a chance for the second player to take the lead.

Figure 4.2: Boxplot of game duration (in turns) by winner in the original Risk map



Source: The Author

Given the game's notable advantage towards the first player to move, our ideal values for the criteria in our experiments are those present in Table 4.2. The ideal value of C_{adv} in our evaluation process is 0.5, incentivizing a balanced game. We establish our T_{pref} as 16, the median of the duration of the games where the second player won, and C_{dur} is set at zero. C_{lc} and C_{dra} are set at 0.1 to incentivize some degree of lead change and drama in the game. When comparing to the original average values in the table below, we consider the average turn duration as the T_{pref} of the original game parameters, and since we had no T_{pref} defined when running the experiments, there was no way to calculate C_{dur}.

Table 4.2: Values used for game evaluation

Metric	*Ideal Value*	*Original Average Values*
C_{adv}	0.5	0.252
C_{dur}	0	Not applicable
T_{pref}	16	15.558
C_{lc}	0.1	0.074
C_{dra}	0.1	0.064

Source: The Author

4.2 Generation of New Games

Using our pipeline implementation to generate game variants described in Section 3.2.3, we created 80 new valid game variants for the simplified Risk. To accomplish it, we created five different maps that, together with the original map, defined our initial population. We used our fitness function with the ideal parameters defined in Table 4.2 to calculate the game's evaluations, playtesting each game 200 hundred times to gather the metrics. When checking the validity of the generated game variations, the pipeline discarded 7 games, resulting in approximately 8% of all generated game variants. The generation of each game variant took around 30 seconds for each iteration, resulting in approximately 45 minutes of execution. Of all 80 valid game variants generated (out of all 87 generated games), the rest of this subsection will discuss the best and worst performing individuals.

The worst performing game had a small 10-territory map with three continents. Table 4.3 shows us the measured metrics from this game across all 200 playtests. This variation's Advantage (C_{adv}) indicates an even more unbalanced game than the original game map, the value of 0.185 indicates that in 81.5% of the games, the first player to move won. Figure 4.3, shows that games where the second player won were, in the median, longer than those where the first player won. Although the Lead Change (C_{lc}) decreased, the amount of Drama (C_{dra}) in this playtest increased compared to the original one. This indicates that lead changes occurred less frequently but were more impactful.

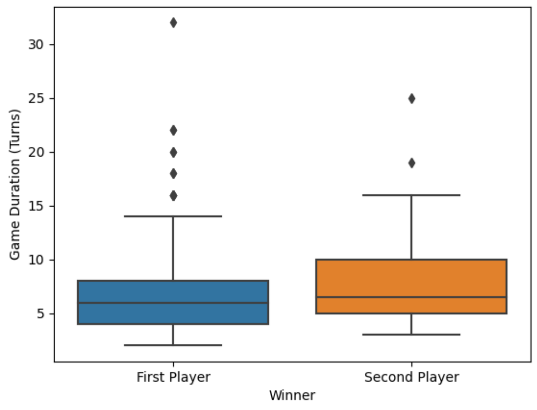
The best performing game had a 51-territory map with six continents. Table 4.4 shows us the measured metrics from this game across all 200 playtests. All metrics are closer to the ideal metrics used in the evaluation in comparison to the original map. With

Table 4.3: Metrics for the worst performing game variation

Metric Measured	*Average Value*	*Standard Deviation*
Advantage (C_{adv})	0.185	Not Applicable
Drama (C_{dra})	0.194	0.188
Lead Change (C_{lc})	0.053	0.116
Average Duration	6.965	4.706

Source: The Author

Figure 4.3: Boxplot of game duration (in turns) by winner in the worst performing game



Source: The Author

an Advantage (C_{adv}) of 0.32, the first player won 68% of the 200 games playtested. Both in the Average Duration and Figure 4.4, we can see that games in this map were more prolonged than in the other two variants analyzed here. Following the same trend of the other games, the second player's wins are, in the median, one to two turns longer than those of the first player. Both the Lead Change (C_{lc}) and the Drama (C_{dra}) values were closer to the ideal values from Table 4.2 as well.

Table 4.4: Metrics for the best performing game variation

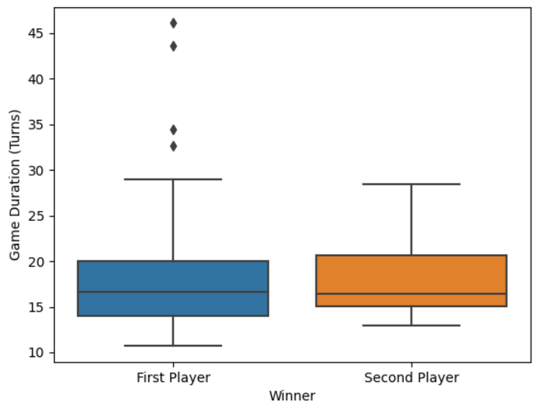
Metric Measured	*Average Value*	*Standard Deviation*
Advantage (C_{adv})	0.320	Not Applicable
Drama (C_{dra})	0.109	0.184
Lead Change (C_{lc})	0.087	0.071
Average Duration	17.308	5.115

Source: The Author

4.3 Discussion

We can examine the effectiveness of evolutionary game design in the context of game variants by analyzing the game variations generated and the measurements of criteria and game duration we extracted from them. The simplified Risk heavily favors the

Figure 4.4: Boxplot of game duration (in turns) by winner in the best performing game



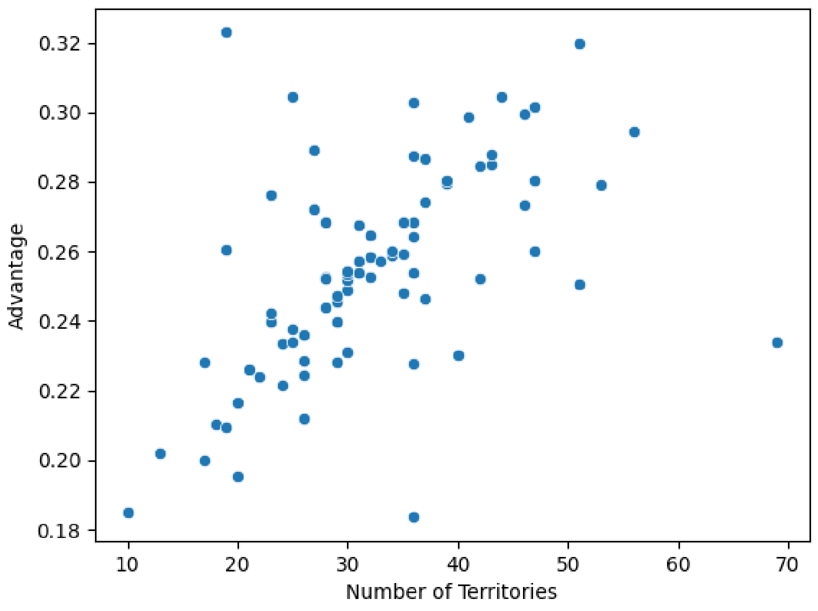
Source: The Author

first player to move, even with the modifications we made to amortize this advantage. In all game variants generated by the pipeline, this advantage to the first player is present, indicating that the game's base rules may be the reason for this imbalance.

Some of the game variants generated by the pipeline were able to reduce this first-player advantage. In the most balanced individual generated by the pipeline, the first player won 68% of the games playtested. In contrast, in the original game's map playtests, the first player won 74.8% of the games. In the individual with the lowest fitness in the pipeline, the first player won 81.5% of the games playtested. These results validate that we can generate more balanced variants for the simplified Risk, but some problems in the game design that originate from the base game's rules may not be solvable using only it.

Analyzing the individuals generated and their playtest's results, we can see that longer games tend to give the second player more chances to win. We can explain this trend by comparing a hypothetical first turn of a player on a map with a low amount of territories and a map with a high amount of territories. In smaller maps, the impact of attacking the enemy first and capturing these territories is higher since each territory in this small map counts as a proportionally bigger control of the whole map. In a larger map, this impact is diminished by the number of territories the second player will still control after the first player's turn. Our results reinforce this hypothesis: Figure 4.5 shows us the relation between the number of territories of an individual and its calculated advantage. Although some of these particular values may be skewed by the innate randomness of the automated playtesting, we notice that smaller maps generate more unbalanced games while larger maps do the opposite.

Figure 4.5: Scatterplot of advantage by number of territories for game variation



Source: The Author

5 CONCLUDING REMARKS AND FUTURE WORK

This work presented a pipeline for the generation of Risk variants through evolutionary game design. We have implemented and validated this pipeline by generating variants of a simplified version of the board game Risk.

In our evaluation, the original game's parameters showed an unbalanced nature, where the first player to move won 74.8% of the games playtested. By generating new game variants and evaluating them through automated playtesting, our implemented pipeline created a variant that mitigated this advantage, where the first player to move won only 68% of the games playtested. These new game variants also varied in length of duration, where games ranged from a median of 6 turns per game to a median of 17 turns per game. Other metrics analyzed showed only a negligible variation between game variants, indicating that they are more related to the game's fixed rules than the game's parameters

Our experiments validated that we can generate new Risk variations through our proposed pipeline. However, we can observe that there are limitations regarding how much impact this generated variant can have on the game's fixed rules. Since we have no control over the rules of how combat works, no game variation can modify the encouragement of aggressive behavior from players, and this can lead to immutable unbalance in the game.

During the development of this pipeline, many ideas that were out of the scope of this work were raised. We point out some of these below as propositions for future work.

An essential aspect for validating our game variants generated through evolutionary game design would be playtesting them with human players. This would require groups that would play different game variations, including the original game's parameters, and a survey with these players regarding perceived balance, game length, and other aspects defined by our proposed criteria.

An interesting development for future work would consist of extending an existing general game purpose framework and implementing the evolutionary game design pipeline. The TAG framework (GAINA et al., 2020), for example, already supports the definition of different game variants through different parameters for each game. It already implements a simple playtesting pipeline that gathers metrics equivalent to our viability criteria. Extending an existing framework for generating new game variants through evolutionary game design would allow us to test different games and agents already implemented in the framework without implementing them from scratch.

Adding new criteria in the evaluation would also be interesting for creating variants with specific qualities. We currently only use four criteria, and of these four, two showed negligible variation in all game variants, indicating that they may be more connected to the game's base rules. Our current criteria are based on Browne's work, but since we are creating variants from a specific game, we can create new criteria specific to this game. Criteria such as the number of continents, number of territories, the average number of troops, or average recruited units per player would be a more direct approach to make these variants have specific desirable qualities.

Much of the current pipeline revolves around the performance of the agents playtesting the game. How these agents play the game influences all the criteria measurements and can be a strong bias in the generation of new game variants. An essential step in future work would be creating more competent agents for the game and creating different agents that emulate a player's actions and objectives. Both Silva et al. (2017a) and Mugrai et al. (2019) show us that we can use different agents to emulate the behavior of human players in a game.

Lastly, an exciting product that could come from this work is the usage of evolutionary game design to create a new board game with the usage of procedural content generation in its design. This approach is not novel in electronic games and has been experimented with in trading card games like KeyForge[1], where each deck is procedurally generated. With games that implement this in their design, the players have a more proactive role while playing. Games with these concepts integrated into their designs bring forth new emerging strategies in each new game, and players must figure them out as they play.

[1]KeyForge is a card game published by Fantasy Flight Games where decks are created procedurally and cannot be modified by players.

REFERENCES

BAKER, J. E. Reducing bias and inefficiency in the selection algorithm. In: **Proceedings of the Second International Conference on Genetic Algorithms on Genetic Algorithms and Their Application**. USA: L. Erlbaum Associates Inc., 1987. p. 14–21. ISBN 0805801588.

BLOMQVIST, E. **Playing the Game of Risk with an AlphaZero Agent**. 2020.

BROWNE, C. **Evolutionary Game Design**. Springer London, 2011. (SpringerBriefs in Computer Science). ISBN 9781447121794. Available from Internet: <https://books.google.com.br/books?id=tn2fE9USzZkC>.

BROWNE, C.; MAIRE, F. Evolutionary game design. **IEEE Transactions on Computational Intelligence and AI in Games**, v. 2, n. 1, p. 1–16, 2010.

BROWNE, C. B. et al. A survey of monte carlo tree search methods. **IEEE Transactions on Computational Intelligence and AI in Games**, v. 4, n. 1, p. 1–43, 2012.

FRAENKEL, A. S. Complexity, appeal and challenges of combinatorial games. **Theoretical Computer Science**, 2004. ISSN 0304-3975. Available from Internet: <https://www.sciencedirect.com/science/article/pii/S0304397503005917>.

FULLERTON, T.; SWAIN, C.; HOFFMAN, S. **Game Design Workshop: Designing, Prototyping, and Playtesting Games**. CMP Books, 2004. ISBN 1578202221. Available from Internet: <https://www.sciencedirect.com/book/9780240809748/game-design-workshop>.

GAINA, R. D. et al. Design and implementation of TAG: A tabletop games framework. **CoRR**, abs/2009.12065, 2020. Available from Internet: <https://arxiv.org/abs/2009.12065>.

GENESERETH, M.; LOVE, N.; PELL, B. General game playing: Overview of the aaai competition. **AI Magazine**, v. 26, n. 2, p. 62, Jun. 2005. Available from Internet: <https://ojs.aaai.org/index.php/aimagazine/article/view/1813>.

HOLMGÅRD, C. et al. Automated playtesting with procedural personas with evolved heuristics. **IEEE Transactions on Games**, v. 11, p. 1–1, 02 2018.

KOZA, J. R. **Genetic Programming: On the Programming of Computers by Means of Natural Selection**. Cambridge, MA, USA: MIT Press, 1992. ISBN 0262111705.

MUGRAI, L. et al. Automated playtesting of matching tile games. **CoRR**, abs/1907.06570, 2019. Available from Internet: <http://arxiv.org/abs/1907.06570>.

PEREZ, D. et al. Rolling horizon evolution versus tree search for navigation in single-player real-time games. In: . New York, NY, USA: Association for Computing Machinery, 2013. (GECCO '13), p. 351–358. ISBN 9781450319638. Available from Internet: <https://doi.org/10.1145/2463372.2463413>.

POLITOWSKI, C.; PETRILLO, F.; GUÉHÉNEUC, Y.-G. **A Survey of Video Game Testing**. arXiv, 2021. Available from Internet: <https://arxiv.org/abs/2103.06431>.

SALEN, K.; ZIMMERMAN, E. **Rules of Play: Game Design Fundamentals**. MIT Press, 2003. ISBN 9780262240451. Available from Internet: <https://mitpress.mit.edu/9780262240451/rules-of-play/>.

SHAKER, N.; TOGELIUS, J.; NELSON, M. J. **Procedural Content Generation in Games: A Textbook and an Overview of Current Research**. [S.l.]: Springer, 2016.

SILVA, F. de M. et al. Ai-based playtesting of contemporary board games. In: **Proceedings of the 12th International Conference on the Foundations of Digital Games**. New York, NY, USA: Association for Computing Machinery, 2017. (FDG '17). ISBN 9781450353199. Available from Internet: <https://doi.org/10.1145/3102071.3102105>.

SILVA, F. de M. et al. Ai-based playtesting of contemporary board games. In: **Proceedings of the 12th International Conference on the Foundations of Digital Games**. New York, NY, USA: Association for Computing Machinery, 2017. (FDG '17). ISBN 9781450353199. Available from Internet: <https://doi.org/10.1145/3102071.3102105>.

THOMPSON, J. Defining the abstract. **The Games Journal**, 2000.