Evolvestone: An evolutionary generator of balanced digital collectible card games

Tiago Zaidan[1]* Luís Fabrício W. Góes[2]

[1]TDZ Games, Research and Development Department, Brazil
[2]PUC Minas, Computer Science Department, Brazil

Abstract

Automated game generation has become desirable to keep up with the ever increasing demand for new and fun digital games. This is a challenging task that requires a huge amount of creativity from different domains, such as game design, art, programming and audio. In recent years, computational creativity on games has been handling this task by studying and proposing several systems capable of producing, if not an entire, at least part of a game content, that is novel and playable. However, no matter how different these games are, the nitty-gritty of their success relies on their balance. For example, a game cannot be too hard or too easy, otherwise a player might lose his interest on it. This paper proposes Evolvestone, an evolutionary generator of balanced digital collectible card games (DCCG), inspired on Hearthstone. It creates Hearthstone-like card games by exploring the space of parameters, building sets of new cards and applying different metrics to ensure game balance. Our main contributions are a genetic algorithm-based system that creates new card games, a full card game simulator and the proposal of balance metrics. Compared to a random approach, results showed that Evolvestone creates card games that are up to 51% more balanced.

Keywords: Computational creativity, Digital games, Digital collectible card games, Game rules, Game evaluation metrics, Evolutionary algorithm, Game balance.

1 Introduction

Computational Creativity is the field which studies the creation of systems that are capable of producing creative artifacts that are new, useful and of quality [23, 5, 10, 2, 1, 22, 7, 3, 18]. In the last years, this research area experienced great advances with the development of systems that are capable of telling stories, composing music, writing poetry and creating pieces of art [5]. In particular, automated game generation is a fertile territory for creativity since digital games are very interdisciplinary and interactive [10, 12, 11].

A digital game is a system in which players engage in an artificial conflict, defined by rules, that results in a quantifiable outcome [21, 11, 17, 6]. There are several game genres, among them the digital collectible card games (DCCG) [19]. Hearthstone: Heroes of Warcraft [1] and Magic: The Gathering [2] are examples of DCCG that have millions of players. These games have a lot of similarities which enables to define a basic common set of parameters for the generation of new DCCG. Although several different DCCG can be automatically created by manipulating these parameters, they should be also balanced, otherwise its abstinence can be considered harmful to the game quality or value [12]. Another important factor to be considered about the balance of a game is that all the players must have the same chance to win, assuming that they have the same skills [11].

The main goal of this paper is to propose and implement an evolutionary generator of balanced digital collectible card games, called Evolvestone, which creates Hearthstone-like card games by exploring the space of parameters, building sets of new cards and applying different metrics to ensure game balance. It is important to note that having in mind the requirements for an artifact to be creative, the balancing of a game ensures only its quality. The main contributions of this paper are:

- The implementation of a genetic algorithm capable of generating balanced digital collectible card games.
- The development of a digital collectible card games full-simulator.
- The combination of three metrics to measure how much a game is balanced.

This paper is organized as follows. Section 2 presents the background, elucidating the main concepts and related work. Section 3 presents the Evolvestone. Section 4 presents the experimental methodology and results. Finally, Section 5 concludes and discusses future work.

2 Background and Related Work

Automatically generated games are commonly created by evolutionary algorithms [13]. These algorithms are very efficient to solve optimization problems with a large solution space, such as the creation of new games [4]. They are guided by a fitness function which is used to evaluate the quality of each possible solution.

In card games, game balance is an essential feature to make it playable. Thus several metrics have been proposed to be used as part of fitness functions. For example, the difference between the number of wins of different players in simulated games is quite often used as a standard balance metric [13, 8]. However, the players winning difference is not enough to ensure that a game is fully balanced, it is necessary to combine other game balance metrics related to uncertainty and changes in leadership [8]. Uncertainty aims to delay the moment that a player gets a big advantage, it can be measured, for example, by the players' life difference. On the other hand, the change in leadership aims to measure how many times

*e-mail: tiagozaidan@gmail.com

[1]available at us.battle.net/hearthstone/en
[2]available at http://magic.wizards.com/content/download

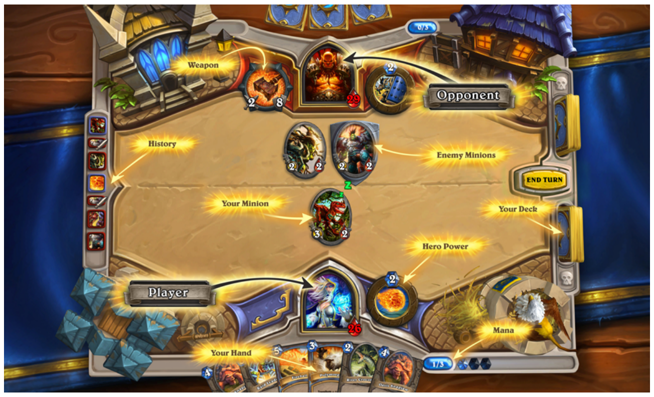

Figure 1: Hearthstone battlefield example[3].

happened a change of the winning player, in order to maximize that value [12].

Although balance makes a game playable, it does not necessarily makes a game interesting to be played. For this reason, other metrics related to idleness and game length can also be used. Idleness measures how many turns occurred no action, that is, the player neither made a move or played a card. A game that has a lot of turns that players can not perform any actions, tends to be less interesting. In addition to it, the metric game length defines the ideal number of turns for a game. A game that is too long or too short might not be so interesting. For this last metric, it is defined an ideal number of rounds and it aims to get as close as possible to that number [8].

In the computational creativity field, automated game generators have been developed specifically to create new and valuable card games [12, 8]. In particular, [8] proposed a system that generates complete card games using evolutionary algorithms with some of the aforementioned metrics. The authors used a context-free grammar to describe and generate versions of known games such as Blackjack and UNO. The generated games were playable, balanced and rewarded the player with greater skills.

Another important work is the generation of novel balanced rules for the card game Dominion proposed in [12]. The authors' main focus was the balancing of different strategies, and for that, it was created three artificial players with different skills capable of playing the game's simulator. In order to generate balanced card sets was used a genetic algorithm that evolves a card set based on three fitness functions with the goal of making the game more interesting. According to those authors, this method could be used more widely to automate the game design and the balancing of other games.

In [14], a recurrent neural network (RNN) was used to generate creative cards for the card game Magic the Gathering. The RNN was able to generate even new effects and most cards were grammatically correct.

Finally, in [9], a computational creativity system, called HoningStone, was proposed. It automatically generates creative card combos for *Hearthstone* based on the Honing theory of creativity. HoningStone used a creativity metric based on surprise and efficiency to generate and evaluate the generated combos. Differently, our work creates new full *Hearthstone*-like card games based on balance metrics.

[3]Source: http://hearthstone.gamepedia.com/Battlefield

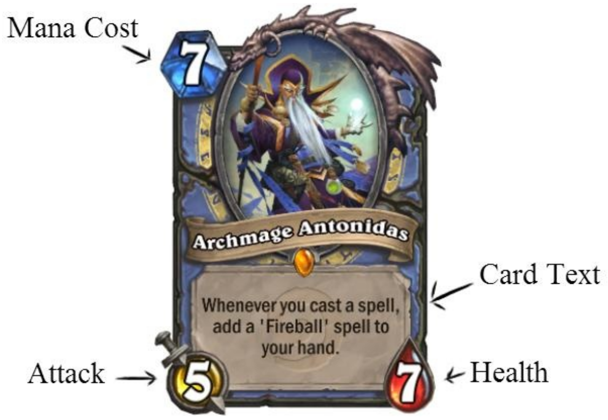
Mana Cost
7
Archmage Antonidas
Card Text
Whenever you cast a spell, add a 'Fireball' spell to your hand.
Attack
5
7
Health

Figure 2: Hearthstone card example[5].

3 EVOLVESTONE

Evolvestone is an evolutionary generator of balanced digital collectible card games[4]. The chosen game parameters for the generation of new DCCG's was inspired mainly in Hearthstone, since it has a simple set of rules. In this section, we present Hearthstone and the selected game parameters to build Hearthstone-like games. Then we introduce our full DCCG simulator, the balance metrics and the genetic algorithm guided by those metrics.

3.1 Hearthstone

Hearthstone is a DCCG in which players compete in one versus one turned based matches until one of the players is defeated [9]. In each turn, a player draws a card from his 30-cards deck and plays some of the cards in his hand to the battlefield. The battlefield is where the combat happens. It can take up to seven cards of each player at the same time. The cards that are not destroyed in the current turn, usually stays in the battlefield for the next turn. Figure 1 shows all the components of a battlefield.

There are three types of cards in Hearthstone, minions, spells and weapons. These cards can have attack points, health points, mana points and effects. The attack points refer to the amount of damage that a card can cause and the health points represent the amount of damage a card can take until it is destroyed. The mana points represents the amount of resources (mana crystals) that a card needs to be played in the battlefield. The card's effects are rules that are triggered by some game's event, for example, a card being played. Figure 2 shows a Hearthstone card example, where the "Card Text" represents the card's effect.

Once a games starts, each player has 30 life points and one is defeated when his life points gets down to 0. In his turn, each player can play any card in his hand to the battlefield. He also draw one card from his deck on each turn. The limitation of the amount of cards a player can play into the battlefield is dictated by the mana pool. This pool starts with only 1 mana crystal and it increases by 1 per turn up to 10 mana crystals.

[4]available for download at https://github.com/tzaidan/evolvestone

[5]Source: http://www.digitaltrends.com/gaming/hearthstone-strategy-guide

3.2 Game Parameters

In order to develop Evolvestone, the first step was the definition of which parameters would be used to determine the basic structure of digital collective card games. Those parameters and its range of values were defined based on a study of Hearthstone. Table 3.2 shows the chosen parameters and their range of values.

Parameters are divided in two main groups: game_params (1 to 10) and cards_params (11 to 17). The game_params are related to the rules and thresholds of the game, while the cards_params are related to the cards attributes and effects. Among the possibilities of values for cards_params, the value 0 means that the parameters won't be used, with exception for parameter card_last, that values from 0 to 4 and represent the amount of rounds a card will last. For the game_parameters, their values are the same represented in the game. For example, suppose a game in which the value for parameter life_points is 30. It means that in this game both of the players will have starting life points equal to 30.

The game_params from 1 to 8 influence both players. The parameters extra_mana and extra_cards are exclusive to player 2. Those parameters have the goal to give some advantage to player 2, in order to minimize the impact of the player 1 be the first to play.

Parameters	Description	Values
1. max_mana	Max amount of mana	5-14
2. mana_rate	Mana rate	1-3
3. life_points	Starting life points	16-52
4. max_card_deck	Max no cards in the deck	18-45
5. max_card_hand	Max no cards in player's hand	5-14
6. max_card_field	Max no cards in the battlefield	5-14
7. withdraw	Deck withdraw rate per turn	1-3
8. cards_1st_turn	No cards withdraw in the first turn	3-12
9. extra_mana	Extra mana for player 2 in first turn	0-9
10. extra_card	Extra cards for player's 2 first draw	0-4
11. max_attack	Max card attack	4-22
12. max_health	Max card health	4-22
13. attack_twice	Attack twice per turn	0-1
14. attack_turn	Attack in the same turn it was played	0-1
15. must_attack	Enemy player must attack this card	0-1
16. card_last	This card will last at max X turns	0-4
17. card_cant	This card can't be attacked	0-1

Table 1 - Parameters and their range of values.

Once defined the values for the parameters card_attack and card_health, the attack and health of the card is randomly generated in the interval 1 to the chosen value. Therefore, regarding these parameters, each card can be different, so players will have different decks.

In order for parameters 13 to 17 not to be the same for every card, it was adopted the use of cards subsets. The game cards were divided in 3 subsets of the same size. For example, if each player has 30 cards in their deck, only the first 10 cards might have the parameter attack_twice with the value 1. It means that in the first 10 cards, each player will be able to attack twice per turn with those cards. However, cards 11 to 20 have the parameters attack_turn and must_attack enabled, it means that those cards will have the rules that those 2 parameters represent, etc. The subsets goal is to bring a more dynamic behavior to the game, because if all the cards have the same parameters 13 to 17 values, the game might be repetitive. Each subset will have its own parameters 11 to 17 values.

In order to determine the mana cost of each card, in this paper, we propose a simple formula that takes into account only the cards attack and health. All game cards use the same formula, so the bigger the attack and health, the bigger is the mana cost. The mana cost is defined in Equation 1, where, atk and hth represent respectively the value of parameters max_attack and max_health.

$$mana = (atk + hth)/2 \tag{1}$$

3.3 Full DCCG Simulator

In order to test and evaluate DCCG games generated by Evolvestone, a full DCCG simulator was implemented that simulates all of the defined parameters. The simulator was written in JAVA and is only executed in batch mode. Each class of the simulator represents an existent object in a DCCG. For example, there are classes that represent players, cards, decks and battlefields. The simulator's main class is the gameplay. It works as a judge and structures the players iteration with the rules of the game. These rules are implemented in the class rules, which describes the behavior of each structural limitation of the game. All possible players moves are implemented in the turn method, within the gameplay class. On every turn, each player draws a card from its deck, gains mana points, plays cards in the battlefield, attacks the enemy and then verifies if he is dead. The class players has a deck, cards in hand and cards in the battlefield. It also has variables that control the amount of life and mana.

The simulator's input is a list containing the parameters as shown in Table 1 with a fixed value within each parameter's range and the number of games to be simulated. The simulator's output is a log containing all the information that feed the genetic algorithm. Figure 3, shows the diagram of the simplified simulator's execution flow.

As the attack and health points of the cards are generated randomly within a defined interval, for each simulated play, new cards and decks are generated for each player to avoid favoring a particular player.

The artificial players (bots) were programmed to represent a human player. There is in the literature [24, 15, 16, 20] several algorithms to implement bots that can compete against human players. In particular, the Monte Carlo tree search (MCTS) is a classical algorithm that uses a heuristic to analyze the most promising moves based on an expanding tree that explores only samples of the search space. However, MCTS is still a computing intensive algorithm. For this reason, in this paper, it was developed a bot that plays only based on the current turn, making it possible to run faster than MCTS and consequently more simulations. As both of the bots play with the same algorithm, the choice of using a simple one will not have a significant impact in the simulated games because of the metrics proposed in this paper.

The developed bot actions are two-fold: to play a card and to attack an enemy. First, it chooses the cards on the player's hand that causes most damage to the enemy, based on the amount of available mana crystals. Then, it seeks to attack directly the enemy with the available cards, respecting the rules of the game and the cards. When attacking, the player can use all the cards in his battlefield, with the exception of cards that have been played in the battlefield in the same turn. It is important to note that when enabled, the parameter attack_turn allows a player to use a card to attack in the same turn it was played in the battlefield (i.e. charge effect).

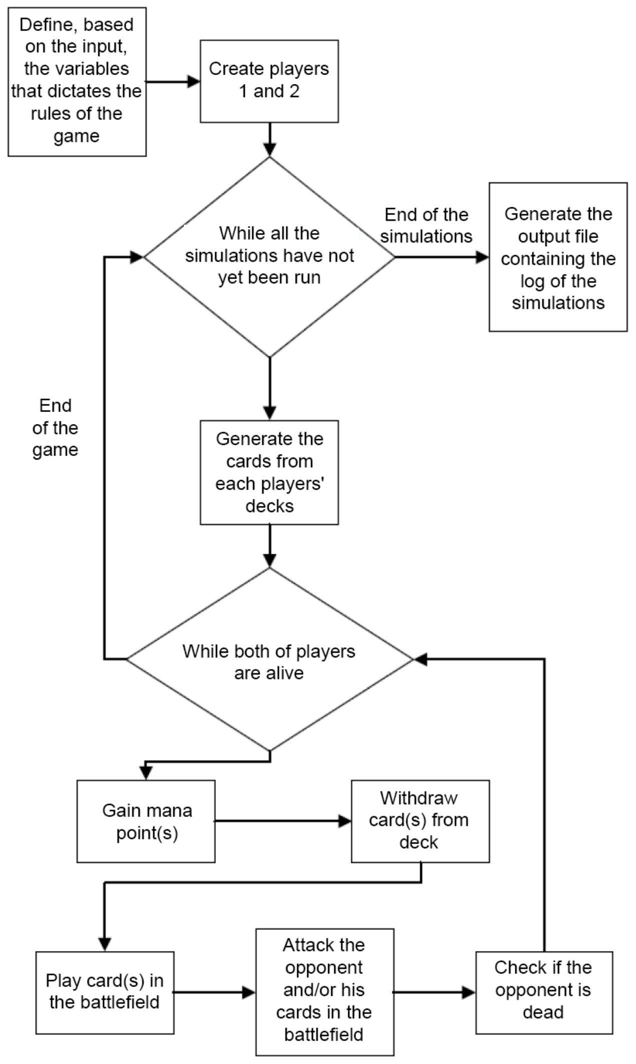
Define, based on the input, the variables that dictates the rules of the game
Create players 1 and 2
While all the simulations have not yet been run
End of the simulations
Generate the output file containing the log of the simulations
End of the game
Generate the cards from each players' decks
While both of players are alive
Gain mana point(s)
Withdraw card(s) from deck
Play card(s) in the battlefield
Attack the opponent and/or his cards in the battlefield
Check if the opponent is dead

Figure 3: Diagram of the simplified simulator's execution flow.

3.4 Balance Metrics

Evolvestone uses three metrics to create balanced games: *win*, *win_rnd* and *win_rnd_life*. These metrics seek to provide fair matches between players with the same skills, so that they have the same probability of winning. Moreover, the three proposed metrics are build on top of each other, so for instance, the *win_rnd_life* metric is composed of the previous two metrics plus an additional one. The metrics are defined as follows:

$$win = |(w1 - w2)| \tag{2}$$

$$win\_rnd = win + |(r - 16)| \tag{3}$$

$$\begin{aligned} win\_rnd\_life = wind\_rnd + |(l1 - 30)| + |(l2 - 30)| \\ + 2 * (pfc1 + pfc2) \end{aligned} \tag{4}$$

where,

- *win*, *win_rnd* and *win_rnd_life* are the proposed metrics;
- *w1* and *w2* represent the percentage of the wins in the simulated games from players 1 and 2 respectively;
- *r* represents the average number of rounds (turns) of the simulated games;
- *l1* and *l2* represents the average final life percentage of player 1 and 2, respectively, in the games they won;
- *pfc1* and *pfc2* represent the average amount of rounds (turns) it took for players 1 and 2, respectively, to play the first card in the battlefield.

The *win* metric measures the difference between both players' winrate in the simulated games, as shown in Equation 2. Complementary, the *win_rnd* metric adds up another term which measures the difference between the average number of rounds in the simulated games and a desirable number of rounds. As mentioned in the previous sections, games that are too short or too long are usually not so interesting. There is no consensus on which it is the optimal average number of rounds. Therefore, it was defined 16 rounds as the average number for each game. It is important to note that this threshold can be also varied.

The *win_rnd_life* also adds up two new terms. The first one is the winner player's final life percentage. It stems from the fact that games where the winner player finishes with his life closer to its initial state, tends to be less interesting, because the game seems to be too easy. Therefore, for a game to become more challenging, the winner player has to face a more balanced match, which means that he has to also lose life during the battle. In this paper, we defined that in a challenging match the winner has to win with an average of 30% of his initial life. This threshold can also be altered, since there is no consensus about it.

The second term added to the *win_rnd_life* metric was related the average number of rounds that each player has to wait before he plays his first card in the battlefield. Games where players spend many rounds without playing his first card tends to be less interesting in general, although there are some specific *handlock* strategies in which a player deliberately hold cards in hand until play them in a certain turn. The number of rounds is normalized by multiplying it by two.

3.5 Genetic Algorithm

The last step in the implementation of Evolvestone was the use of genetic algorithm to generate game instances. It was decided to use the Watchmaker[6] as the genetic algorithm framework. It is written in JAVA and has all the needed functions already implemented. Therefore, it is necessary only to adapt those functions for the problem presented in this paper.

The individuals in the population are represented as a 17-position array, where each position represents a different parameter, that is, each individual represents a possible game. Each position in the array takes in only the value of the parameter it represents. For example, the first position of the array represents the parameter *max_mana* that defines the maximum amount of mana and can have values of the interval 5 to 14. Those individuals are evaluated by a fitness function that is represented by one of the metrics aforementioned. The most adapted individuals are those that minimizes the

[6]Available at http://watchmaker.uncommons.org/

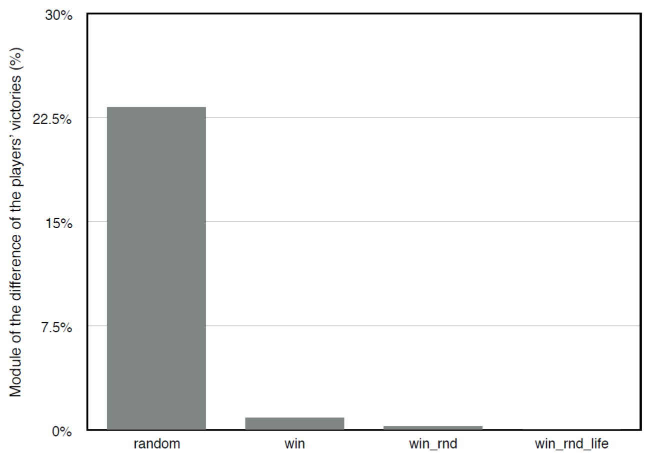

Figure 4: Percentage of wins difference.

fitness function, the best individuals are the ones with the output equal to 0.

It is used elitism in the most adapted individuals, so that the best game instances move to the next generation without suffering any changes in their genetic code. Crossover is applied to the rest of the individuals to generate new individuals in the population. During crossover, two individuals combine the value of each position of their array in order to generate a new individual. Mutation is implemented by randomly changing the value of some position in the array of an individual, becoming a new mutated one. The new generated individuals and the ones that did not change because of the elitism are evaluated again by the fitness function in the next generation. This cycle repeats until an individual that matches the requirements of a balanced game, dictated by the fitness function, is found.

4 Experimental Results

In this section, we evaluate each proposed metric compared to a random approach as a baseline in three sets of experiments, namely: same rules, different rules and parameters individual influence. In the first two experiments, we observe four factors: i) the difference between players wins; ii) the average number of rounds; iii) the winner's final life and iv) the average number of turns before a player plays his first card. They differ only in the fact that in the first one, the rules are the same for both players and in the latter each player plays by different rules. The third experiment was designed to identify which parameters most impact the game balance. In all experiments, the number of simulations per game instance was set to 1000 and the number of games per metric/approach to 10. All results were averaged by a simple arithmetic mean, with a 95% confidence level.

Figure 4 shows the module of the difference of players 1 and 2 wins. It shows that all the three metrics had a value less than 1%. It stems from the fact that all of them includes the *win* metric which evaluates the number of wins between players. On the other hand, the random baseline achieves 22.5%, which is very unbalanced.

Regarding the number of rounds per match, Figure 5 shows that both metrics *win_rnd* and *win_rnd_life* were able to reach the desirable number of 16 rounds. Since the metric *win* and the *random* approach do not evaluate the average number of rounds, both of them did not reach the average of 16 rounds. On average, the *random* approach generated shorter games while the *win* metric generate longer ones.

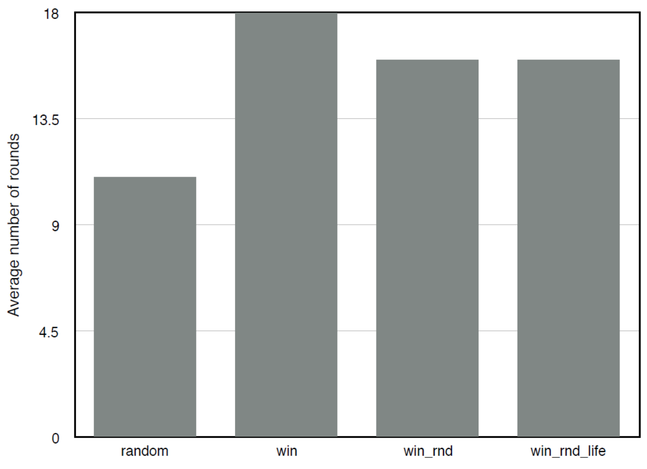

Figure 5: Average number of rounds.

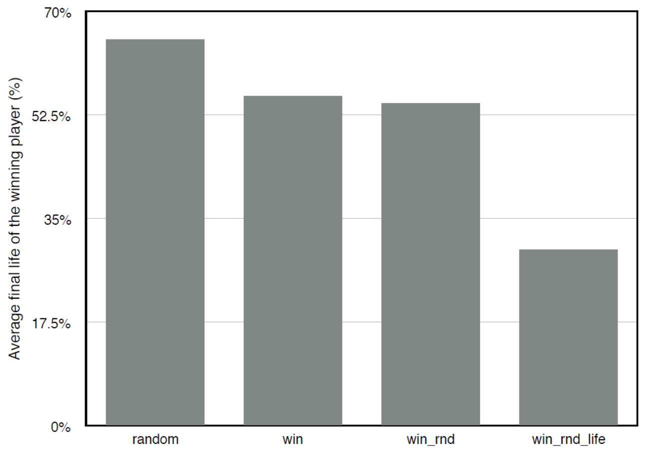

Figure 6: Winning player's final life.

Figure 6 shows the average final life percentage of the winner player. The *win_rnd_life* metric had the value of 29.9%, that is, it was able to reach the determined value of 30%. On the other hand, Figure 7 shows the average number of turns needed for each player to put his first card in the battlefield. It is important to note that the value 0 means that the player was able to put a card in the battlefield in his first turn. It can be observed that the *win_rnd_life* metric was the only one able to minimize this value, so all the players on average are able to put at least one card in the battlefield in their first turn.

An interesting result that emerged from Figures 4 to 7 is that all factors presented on each metric are independent but they need to be together to make a game more balanced. For example, a game in which players have the same average amount of wins does not guarantee that it will be more challenging and fair, since most wins will be easy and straightforward, probably determined by the early

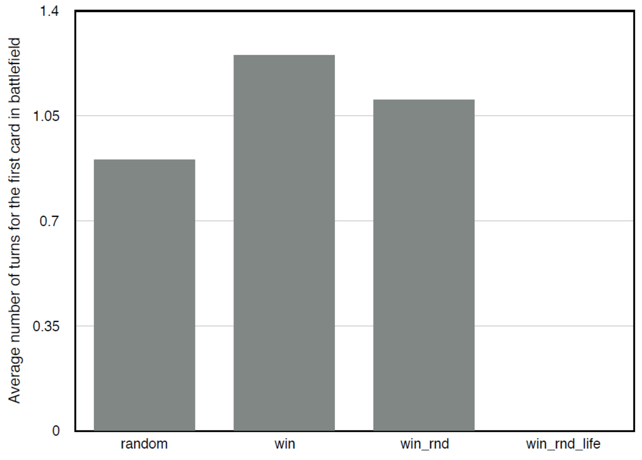

Figure 7: Average number of turns for the first card in battlefield.

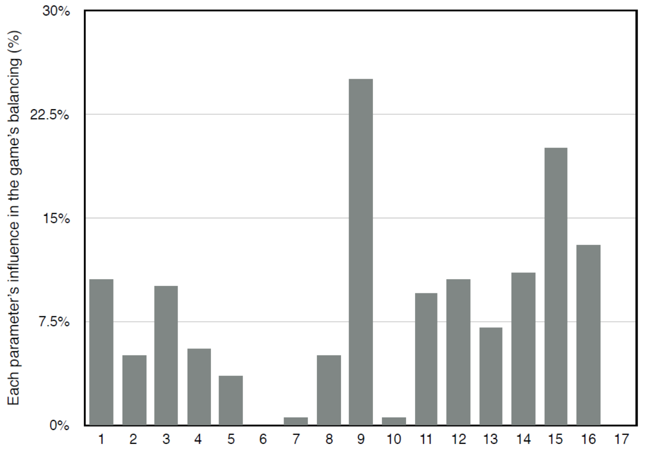

Figure 8: Each parameter's influence in the game's balancing.

game. Thus only the full set of all factors together, represented by the *win_rnd_life* metric, can make a game more balanced and interesting. In summary, the random approach generated games that have an average value of 51% when evaluated by the *win_rnd_life* metric. It means that the games generated by Evolvestone using the *win_rnd_life* metric are 51% more balanced than the games generated randomly.

The second set of experiments was based on a recent game mode created by Hearthstone, called Tavern Brawl. In this game mode, on each week, a new match with different set of rules is generated. In some matches, there is even the possibility that each player plays with a different set of rules, which brings a more dynamic experience to the game. In this paper, in order to generate games with a different set of rules, it was considered that each player can have different values for each of the 17 parameters. It was applied the same process of the first experiment with this single difference. The obtained results were very close to the ones shown in Figures 4 to 7, so they are not presented. This interesting result shows that even with players governed by different rules, it is still possible to generate games that are balanced.

The last set of experiments has the goal of determining how much each parameter impacts the game balance. For this experiment, we used the *win_rnd_life* metric which is the most robust one. In order to determine the influence of each single parameter, we used a single factor experimental design where each parameter is changed from the minimum to its maximum value at a time and compared with its optimal value (the value determined by the genetic algorithm when it generated a game using the *win_rnd_life* metric as fitness function). For example, if the variation of a parameter's value does not differ from the simulation optimal output, that is, the *win_rnd_life* metric output is still the same, this particular parameter has no influence in the game balance. The influence of each parameter was calculated based on the difference between its minimum and maximum values arithmetic mean result on the *win_rnd_life* metric output and the optimal *win_rnd_life* metric output that was close to 0.

Figure 8 shows the percentage of influence of each parameter in the game balance. The parameter 9, *extra_mana*, is the most influential one with more than 20% impact. This parameter determines the player's 2 amount of extra mana in one round. The minimum parameter of this rule does not give any extra mana for player 2, causing that the player 1, that already has the advantage of playing first, increases both his number of wins and his final life in the matches he won. In turn the parameter's maximum number allows the player 2 to play what is equivalent to two turns in a row, giving him a greater advantage.

After *extra_mana*, the parameters 15, 16, 14 and 12, respectively *must_attack*, *card_last*, *attack_turn* and *max_health*, are the ones that influences the most. Those four parameters refer to the *cards_params* and can give a great advantage for one of the players, winning more games faster and with a higher final life. Differently, parameters 1, *max_mana*, and 3, *life_points*, that refer to the maximum amount of mana and life, respectively, do not affect significantly the game balance, since they mainly alter the number of game rounds.

On the other hand, parameters 6, *max_card_field*, and 17, *card_cant*, do not influence the balancing of the games, so it is possible to use any value for those parameters. In particular, *max_card_field* determines the maximum number of cards in the battlefield. This parameter does not have any influence because its parameter's minimum value is 5, which is relativity high. In turn, *card_cant* in which the player's card can not be attacked, does not have any influence because of the bot behavior. As it always seeks to attack the opponent when possible, it will not make any difference the fact the it is not allowed to attack with a card with this effect.

Finally, parameters 7, *withdraw*, and 10, *extra_card*, almost do not influence the game balance. They determine the amount of cards in the first draw and player's 2 extra amount of cards in the first draw, respectively. Those rules do not influence too much because the players start with only 1 mana crystal. It means that at the beginning of the game the players cannot put many cards in the battlefield.

In summary, these final results revealed many insights that can be used in practice when designing balanced card games. For instance, the amount of extra mana given to player 2 showed to have a significant impact in the game balance, while the initial number of card draws does not seem to be so decisive.

5 Conclusion

Despite the last decade's great advance in the field of computational creativity, digital games have not yet been so explored. In the light of this scenario, it was proposed in this paper, the creation of an evolutionary generator of balanced digital collectible card games, called Evolvestone. The balance of Hearthstone-like card games generated by Evolvestone was assured by the use of three metrics. The results showed that the generated games were up to 51% more balanced than games randomly generated.

The main approach for future work is the creation of metrics to evaluate if the generated games are fun to play. For this goal, it is mandatory the creation of an interface so that players can evaluate those games gameplay. This analysis can also confirm if the games are balanced for human players. It could further determine others aspects to be considered for the creation of other metrics. The balance influence of each parameter calculated in this paper can be used to create a new mana formula. Another approach for future work is the creation of new parameters and the use of a bigger parameter's range for the existing parameters. Finally, it can be developed more complex bots and compare their performance to the one used in this paper.

6 Acknowledgment

We would like to thank FIP PUC Minas, FAPEMIG, CNPq and CAPES to support this work.

References

[1] T. Besold. *Computational creativity research: towards creative machines*. Atlantis Press, Amsterdam, 2015.

[2] M. Boden. *The creative mind myths and mechanisms*. Routledge, London New York, 2004.

[3] S. Colton. Creativity versus the perception of creativity in computational systems. In *AAAI Spring Symposium on Creative Intelligent Systems*, Technical Report SS-08-03, pages 14–20, 2008.

[4] S. Colton, A. Pease, J. Corneli, M. Cook, and T. Llano. Assessing progress in building autonomously creative systems. In *International Conference on Computational Creativity (ICCC)*, pages 137–145, 2014.

[5] S. Colton and G. A. Wiggins. Computational creativity: The final frontier? In *European Conference on Artificial Intelligence (ECAI)*, pages 21–26, 2012.

[6] T. M. Connolly, E. A. Boyle, E. MacArthur, T. Hainey, and J. M. Boyle. A systematic literature review of empirical evidence on computer games and serious games. *Computers and Education*, 59:661–686, 2012.

[7] M. Cook and S. Colton. Ludus ex machina: Building a 3d game designer that competes alongside humans. *International Conference on Computational Creativity (ICCC)*, pages 54–62, 2014.

[8] J. M. Font, T. Mahlmann, D. Manrique, and J. Togelius. Towards the Automatic Generation of Card Games through Grammar-Guided Genetic Programming. In *Proceedings of Foundations of Digital Games (FDG)*, pages 360–363, 2013.

[9] L. F. W. Goes, A. R. da Silva, J. Rezende, A. Amorim, C. Franca, T. Zaidan, B. Olimpio, L. Ranieri, H. Morais, S. Luana, and C. A. P. S. Martins. Honingstone: Building creative combos with honing theory for a digital card game. *IEEE Transactions on Computational Intelligence and AI in Games*, PP(99):1–1, 2016.

[10] A. Liapis, G. N. Yannakakis, and J. Togelius. Computational game creativity. In *International Conference on Computational Creativity*, pages 46–53, 2014.

[11] T. Mahlmann. *Modelling and Generating Strategy Games Mechanics*. PhD thesis, ITU Copenhagen, December 2012.

[12] T. Mahlmann, J. Togelius, and G. Yannakakis. Evolving card sets towards balancing dominion. In *IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8, 2012.

[13] J. Marks and V. Hom. Automatic design of balanced board games. In *Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE)*, pages 25–30, 2007.

[14] R. M. Milewicz. Generating magic cards using deep, recurrent neural networks, 2015. Disponvel em http://www.mtgsalvation.com/forums/creativity/custom-cardcreation/612057- generating-magic-cards-using-deep-recurrent-neural.

[15] I. Millington. *Artificial intelligence for games*. Morgan Kaufmann/Elsevier, Burlington, MA, 2009.

[16] I. Millington. *Artificial Intelligence for Games*. CRC Press, 2009.

[17] J. Novak. *Game development essentials : an introduction*. Delmar, Clifton Park, N.Y, 2012.

[18] H. G. Oliveira and A. Cardoso. Poetry Generation with PoeTryMe. *Computational Creativity Research: Towards Creative Machines*, 7:243–266, 2015.

[19] S. Rogers. *Level up! the guide to great video game design*. Wiley, Hoboken, 2014.

[20] N. Sephton, P. I. Cowling, E. J. Powley, and N. H. Slaven. Heuristic move pruning in monte carlo tree search for the strategic card game lords of war. In *IEEE Conference on Computational Intelligence and Games (CIG)*, pages 1–7, 2014.

[21] K. Tekinbas. *Rules of play : game design fundamentals*. MIT Press, Cambridge, Mass, 2003.

[22] L. Varshney, F. Pinel, K. Varshney, A. Schorgendorfer, and Y.-M. Chee. Cognition as a part of computational creativity. In *IEEE International Conference on Cognitive Informatics Cognitive Computing (ICCI*CC)*, pages 36–43, 2013.

[23] O. Vartanian. *Neuroscience of creativity*. MIT Press, Cambridge, MA, 2013.

[24] C. Ward and P. Cowling. Monte carlo search applied to card selection in magic: The gathering. In *IEEE Symposium on Computational Intelligence and Games (CIG)*, pages 9–16, 2009.