LUDUS: An Optimization Framework to Balance Auto Battler Cards

Nathaniel Budijono*[1], Phoebe Goldman*[2], Jack Maloney*[3], Joseph B. Mueller[4,5], Phillip Walker[5], Jack Ladwig[5], Richard G. Freedman[5]

[1] University of Minnesota Twin Cities
[2] New York University
[3] University of Wisconsin-Madison
[4] University of Minnesota
[5] SIFT

budij001@umn.edu, phoebe.goldman@nyu.edu, jmaloney3@wisc.edu,
jmueller@sift.net, pwalker@sift.net, jladwig@sift.net, rfreedman@sift.net

Abstract

Auto battlers are a recent genre of online deck-building games where players choose and arrange cards that then compete against other players' cards in fully-automated battles. As in other deck-building games, such as trading card games, designers must balance the cards to permit a wide variety of competitive strategies. We present LUDUS, a framework that combines automated playtesting with global search to optimize parameters for each card that will assist designers in balancing new content. We develop a sampling-based approximation to reduce the playtesting needed during optimization. To guide the global search, we define metrics characterizing the health of the metagame and explore their impacts on the results of the optimization process. Our research focuses on an auto battler game we designed for AI research, but our approach is applicable to other auto battler games.

1 Introduction

Auto battlers are a new and wildly popular genre of online game. *Auto Chess* released in January of 2019 (Drodo Studio 2019), and within that same month was regularly seeing 70,000 concurrent players (Tack 2019). By June, *Dota Underlords* and *Teamfight Tactics* were among the most popular online games in the world (Grayson 2019). Since then, entries like *Hearthstone: Battlegrounds* (Blizzard Entertainment 2019), *Fire Emblem Heroes: Pawns of Loki* (Nintendo Mobile 2020), and *Storybook Brawl* (Good Luck Games 2021) have further refined the genre.

In an auto battler, a number of players (typically eight) compete to build the best lineup of cards through a number of rounds. Each round consists of a deck-building phase and a battle. In the deck-building phase, each player improves their lineup by selecting cards from a shared pool and arranging their order. In the battle phase, two players' lineups compete automatically. Players who lose too many of these battles are eliminated. Play proceeds in rounds until all but one player are eliminated. The remaining player wins.

Auto battlers are in many ways similar to deck-building collectible card games like *Magic: the Gathering* (Wizards of the Coast n.d.b), *Pokémon* (The Pokémon Company n.d.), and *Yu-Gi-Oh!* (Konami n.d.). In collectible card games, a battle between two decks is called a game. Because it encompasses and affects multiple games, the deck-building process is called the *metagame*.

Deck-building games and metagames are often described as being either *healthy* or *unhealthy*. The health of a metagame depends on a variety of factors, of which we highlight *diversity*. Diverse metagames are those where many different decks/lineups coexist with similar overall win rates.

Metagame balance in deck-building games poses a challenge for game designers because a relatively small number of cards are combined to form a large number of possible decks, and these decks are paired into an even larger number of possible match-ups. *Magic: the Gathering* relies on human playtesting in advance of releasing a set (DeTora 2017), but human playtesting before release has repeatedly failed to identify card designs or sets that lead to unhealthy metagames. Even when it does successfully identify and fix unhealthy metagames, human playtesting requires significant labor and time from a number of skilled players.

In addition, digital card games like *Hearthstone* collect and analyze data after release from consumer play (Zook 2019). While inexpensive, this approach is insufficient on its own because it requires a significant number of consumers to play in unhealthy metagames in order to identify problematic designs.

Healthy and diverse metagames are desirable because they lead to more varied, and therefore more enjoyable, battles. In a homogenous metagame, any two battles are relatively similar. This reduced novelty bores players, who may then reduce their involvement with the game and purchase fewer products. Also, unsatisfied players often share their opinions on social media, discouraging new players from joining the game.

For example, *Magic: the Gathering*'s set *Throne of Eldraine* was criticized for containing a number of overpowered cards. Decks that contained these powerful cards reliably beat decks without them. High-level players quickly converged on a small number of deck designs that best uti-

*These authors contributed equally.

lized the powerful *Throne of Eldraine* cards. On at least one occasion, a tournament was canceled due to lack of interest (Triske 2019). Wizards of the Coast, the company responsible for *Magic: the Gathering*, banned a total of six cards from *Throne of Eldraine* in response (Wizards of the Coast n.d.a; Duke 2019, 2020b,a). Even after the bans, high-profile professional players have complained on social media about *Throne of Eldraine*'s impact on the *Magic: the Gathering* metagame (Scott-Vargas 2020).

In this paper, we introduce LUDUS, a framework to reduce the cost of playtesting without exposing consumers to unhealthy metagames by automating the balancing process. Section 2 discusses other research related to auto battlers and deck-building metagames. Section 3.1 develops an auto battler game we designed to facilitate artificial intelligence (AI) research and presents an algorithm that efficiently and accurately approximates the average win rates of all the lineups in a metagame. Section 3.2 defines three metrics that evaluate the diversity of a metagame, and we theorize how game designers could define more precise metrics for their own games. Section 3.3 explains how we use a genetic algorithm to balance a metagame. Section 3.4 explores how game designers can use LUDUS and these metrics for insights about their current card designs. Section 4 presents experimental results evaluating the sampling-based approximation described in Section 3.1 and the genetic optimization described in Section 3.3. Section 5 reviews the implications of our experimental results. Section 6 considers possible extensions to our work.

2 Related Works

As the auto battler genre is still new, we were only able to find one other work that specifically addresses the use of AI in their design. Xu et al. (2020) focused on autonomously identifying the best lineups that players can construct given the available cards, game rules, and lineups that the game's designers expect to be played the most. [1]. In a two-step process, their method first simulates play between randomly generated lineups and the designers' expected lineups to create a training dataset for a neural network—the model learns a mapping between a given lineup and its estimated win rate against the designers' expected lineups. In the second step, a genetic algorithm searches for lineups that optimize the learned win rate function under various constraints that define lineup construction rules throughout the auto battler game. Constraints account for drafting additional cards between rounds of play, which are not elements of our auto battler game (see Section 3.1 for a description).

Game designers may use the optimized lineups to evaluate whether the cards are balanced and, if not, which cards need revision. Our method instead revises the cards directly, optimizing the viability of the set of possible lineups that players can construct. This alters the initial efforts of the game designers to focus on the rules and some card templates without committing to specific grounded card instances. Because the templates are lifted representations of the actual cards, it is more difficult for designers to identify which lineups are expected to be common in the metagame. However, after our approach provides suggestions for grounded card instances, game designers can apply Xu et al.'s methods for a deeper analysis of the suggested card designs. If the designers have some expected lineups based on the templates alone, then it should be possible to combine our methods. The information available to the game designers would enable them to explore not just which lineups are considered the best, but also how the best lineups change for various card revisions.

Many games that support competitive play over the internet collect data about what and how people are playing. Game designers can take advantage of various data science approaches to find trends in the logged data that inform them about how to balance the game (Nguyen, Chen, and Seif El-Nasr 2015). In the virtual collectible card game Hearthstone, clusters of similar deck compositions implicitly describe archetypes that are currently popular in the metagame. Designers can use this information to decide if some archetypes are too common (implying they might be too powerful) and respond through updates to card descriptions, releasing new cards that counter the overused archetypes, or releasing new cards that support the underused archetypes (Zook 2019). This organizes human-operated playtesting at macro-scale where designers can iterate on their game between updates.

Because human-operated playtesting is expensive resource-wise and rarely exhaustive enough to identify all points of concern, automating the playtesting process can speed up the number of games played and discover edge cases humans might not consider. For deck-building games where the players have agency and outcomes are nondeterministic (shuffled decks, random outcomes of effects, etc.), the space of possible deck combinations and game states can be immeasurable. Following in the footsteps of DeepMind's work on AlphaGo (Silver et al. 2016), Stadia trained a deep learning model through many games of self-play in order to develop a function that could evaluate the quality of a game state for a deck-building game (Kim and Wu 2021). Their automated game-playing agent then competed against itself with designer-constructed decks in order to compare the overall decks' performance, accounting for the nondeterministic outcomes through many games as statistical samples. A game designer may manually inspect the outcomes of all the matches to determine whether any revisions to the cards are necessary.

Bhatt et al. (2018) investigated the deck-building aspect of Hearthstone with genetic algorithms evolving the decks to optimize defeating a specified opposing deck. Their game-playing agents used a greedy, myopic search algorithm as an experimental control while the evolving decks were the experimental cases. The resulting evolutionary paths yield insights into the diversity of the metagame. In contrast, Zook, Harrison, and Riedl (2019) altered the parameters of Monte Carlo Tree Search to study how different types of players performed with fixed decks in their own collectible card game. It revealed balance issues from the perspective of the game's rules, such as a clear advantage to the player with the first turn. To investigate more specific scenarios, Jaffe et al. (2012) enabled designers to specify restrictions on player parameters and available cards. Their simulator ran multiple

[1] Their specific auto battler uses 'piece' instead of 'card.'

games and summarized the results with respect to checking various features of the win percentages and play logs. Chen and Guy (2020) considered both simulating multiple games and training a deep learning model to assess the metagame as they procedurally generated new cards using grammars.

3 Methods

To achieve the goals outlined in the introduction, LUDUS contains an auto battler game with a number of special cards with extra mechanics and a basic 'vanilla' card that only engages in standard combat without extra mechanics. The details of these cards and their mechanics are explained below in Section 3.1. The health and attack values of all cards can be adjusted as well as some values pertaining to the extra mechanics (such as how much damage to deal to an opponent's card when a card dies). Multiple cards of the same type that have different attack, health, or other values can be considered in the same game.

Beyond the auto battler itself, LUDUS also contains an optimization framework to find configurations of available cards that are balanced, fair, and elicit desired play experiences. These criteria are obviously quite subjective; so game designers can easily plug new metrics into the existing framework as they develop new ways of capturing their goals and preferences describing the metagame. This paper explores a few options as well.

3.1 Playtesting Simulator

To serve as a data source for quantitative analysis, we create a playtesting simulator for an original auto battler game we designed. This simulator is designed for research purposes, allowing the user to create cards with new mechanics and run tournaments between arbitrary lineups.

All cards are equally available to all players, and there are no restrictions on the number of copies of a card allowed in players' lineups or in the entire game. Some auto battlers have a drafting process in which cards are selected or purchased from a pool, but drafting and/or deck-building take different forms across different games (Blizzard Entertainment 2019; Good Luck Games 2021; Nintendo Mobile 2020). We do not explore variants of these processes in this paper, but we will consider them in future work.

Card Definition Cards have positive integer statistics including health points, attack, and parameters pertaining to a special mechanic. In our game, such special parameters include the following:

Explosion Damage When the card dies, it damages each of the opponent's cards by this many health points.

Heal Amount Prior to entering combat, the card restores this many health points to its player's rightmost card

Attack Growth Per Hit When the card takes damage, its attack increases by this many points.

Explosion Heal When the card dies, it heals each of its player's other cards by this many health points.

Heal Donation Percent When the card would be healed, it instead restores this many percent of the health points to itself and donates the remaining health points to its player's other cards evenly.

Armor Points The card starts with this many armor points. When the card would be damaged, it instead loses an armor point. When the card has zero armor points, it is damaged normally.

Damage Split Percent When the card would be damaged, it instead receives this many percent of the damage to itself and splits the remaining damage evenly to its player's other cards.

Middle Age Prior to entering combat, the card increases its attack by 1 if it has not participated in this many combat phases. After this many combat phases, the card decreases its attack by 1 before entering combat.

Target Age When the card dies, if it participated in this many combat phases, it deals this many damage points to each of the opponent's cards. If the card participated in fewer than this many combat phases, it heals each of its player's other cards by health points equal to the number of combat phases in which the card participated.

Detonation Time When the card has been healed or damaged this many times, the card and its opponent's card both die.

Other card mechanics without parameters include the copying the attack of the opponent's cards (morphing enemies) and swapping its health points and attack whenever its attack is greater than its health points (survivalist).

Match Procedure Gameplay proceeds as follows in the simulator:

1. Each player in a tournament selects an ordered list of cards to be their lineup.
2. Prior to combat, arrange the lineup's cards left-to-right.
3. Each combat phase pits the leftmost surviving card of each player against each other. The cards take damage equal to their opponent card's attack. Cards with 0 or less health points die, and surviving cards are rearranged to be the rightmost card of their respective lineups.
4. Repeat the combat phase until one or fewer players has a surviving card or the maximum number of combat phases has been reached. If both or neither players have at least one surviving card, the game is a draw. Otherwise, the player with a surviving card wins.

Win Rate Approximation We run a round-robin tournament where each lineup plays one match against every other lineup. Unfortunately, for n lineups, this results in $n(n-1)/2 = n^2/2 - n/2$ matches. This quickly becomes prohibitively large to simulate. We developed a sampling-based approximation that runs a subset of the games in order to estimate the win rates of cards and lineups. It randomly partitions the lineups into groups of uniform size and runs round-robin tournaments within these groups. For a group tournament with k groups and n total lineups, this results in an upper bound of $k \cdot (n/k)(n/k-1)/2 = n^2/(2k) - n/2$ matches in the case where n is divisible by k.

3.2 Metrics

Quantitative measures of metagame health are needed to guide card statistic optimization. We present metrics built upon the output data of the playtesting simulator.

The playtesting simulation outputs a data vector v, the *lineup payoff vector*. Each entry $v_i \in [-1, 1]$ represents the average payoff of the lineup at index i during the last round of playtesting in the optimization process. A payoff of 1 indicates a win. A payoff of 0 means a tie, and a loss results in a payoff of -1. The average of these payoffs over the course of the simulated tournament for a lineup make up the entries of v.

We then transform v into another vector of length n that maps a card to the average win rate of the lineups in which the card appears. This vector w, the *card win rate vector*, has entries $w_j \in [0, 1]$, the average win rate of the lineups in which the card at index j appears. We present three metrics that evaluate the uniformity of this distribution of win rates among the cards.

Each of the following metrics operate on the card win rate vector.

Per-card Payoff

$$\mathrm{PCP} = \frac{1}{n} \sum_j |\mathrm{payoff}(w_j)|^2 \tag{1}$$

Since average payoffs are equal to 0 when a lineup a card appears in wins as many matches as it loses, this is a metric to minimize if we want to punish overly powerful or terrible cards. The squared term disproportionately attacks outliers.

Standard Deviation Metric

$$\mathrm{SDM} = \sqrt{\frac{1}{n} \sum_j \left| w_j - \left(\frac{1}{n} \sum_j w_j \right) \right|^2} \tag{2}$$

Minimizing standard deviation of the win rates may avoid cards with drastically higher or lower win rates compared to other cards.

Entropy Metric

$$\mathrm{EM} = -\sum_j \frac{w_j}{\sum_j w_j} \log \left(\frac{w_j}{\sum_j w_j} \right) \tag{3}$$

When the entropy of the win rates is maximized, we approach a uniform distribution of win rates over the cards.

Other Metrics The LUDUS framework is flexible; researchers and designers alike may develop their own metrics corresponding to their own ideas of what constitutes a healthy metagame. For example, a game designer may know from historical data that a certain distribution of win rates over cards or lineups correspond to a period of celebrated parity in the metagame. The designer could conceivably use the Wasserstein distance or Kullback-Leibler divergence between this historical distribution and another distribution as a metric. Minimizing this metric might maintain similar levels of parity through future iterations of card releases. Metrics could also evaluate other measurable elements like average turns per match.

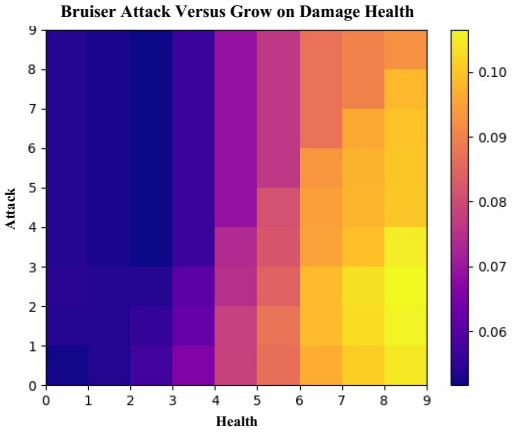

Figure 1: Heatmap showing how changes in the attack of the bruiser card and the health of the grow-on-damage card, while all other card parameters remain fixed, affect the standard deviation metric.

3.3 Optimization

We use the PyGAD python library (Gad 2021) for genetic algorithms. This searches for the optimal values of specified parameters, which may include health points, attack, and special parameters, with respect to the chosen metric. For our experiments, we use the standard deviation metric.

We constrain the genetic algorithm to consider integer values between 1 and 10, inclusive. While hyperparameter tuning may have yielded better results, we chose to run the genetic algorithm with 8 solutions and 4 parents mating per population for 32 generations in each experiment.

3.4 Qualitative Analysis

Another way to understand the landscape of the metagame is through more qualitative methods. We look at two different ways of plotting data about the game results that give insights into the health and stability of the metagame.

The first plot sweeps over two variables, typically of the same card, although not necessarily. This gives a visual representation of the metagame sensitivity under variations of a configuration, which can lead to some interesting insights about the relationships between cards.

Take, for example, the plot in Figure 1 of the value of the standard deviation metric given various values for the attack of the bruiser card and the health of the grow-on-damage card. The bruiser card is a vanilla card with a high attack (although variable in this situation) and one health point. The grow-on-damage card has one attack growth per hit. Initially this card has one attack, and in this situation has a variable amount of health. If the grow on damage card is able to consistently become a powerful threat, it can become too strong and a necessity for high-level play, leading to an unbalanced metagame.

In the figure, we can see that configurations in the upper left corner, where the attack of the bruiser is greater than the health of the grow on damage card, tend to be much more

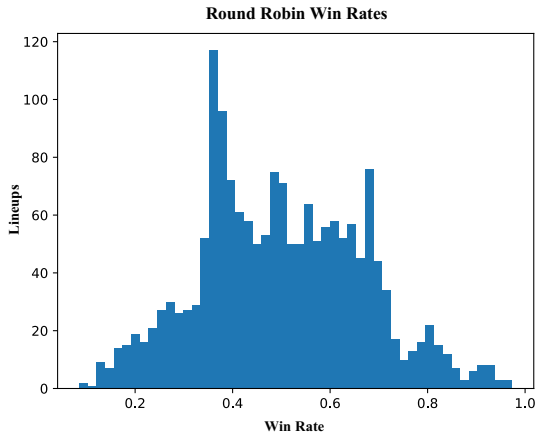

Figure 2: Distribution of win rates before optimization.

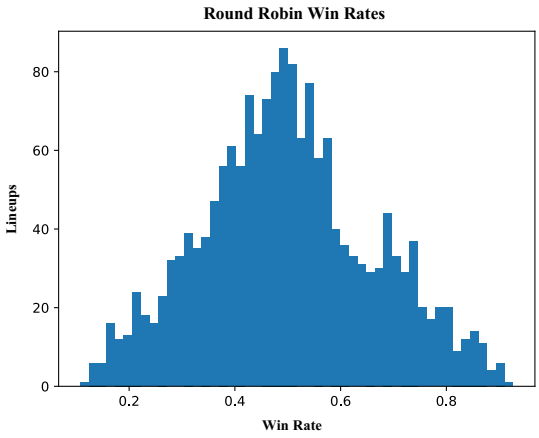

Figure 3: Distribution of win rates after optimization.

favorable than the configurations in the lower right corner, where the opposite is true. This evidence supports the intuitive idea that the bruiser's high attack is a good counter to the grow on damage card because the bruiser card can quickly remove the grow on damage card before it has had enough interactions to grow its damage to a large number. This kind of information can be quite useful to a game designer, who can see very clearly that the inclusion of a high-damage card can improve the metagame dramatically if the grow on damage card is found to be too powerful.

The second plot is a histogram of the lineup win rates after running a tournament amongst all possible lineups. The resulting distribution can have features that may be informative to game designers. For example, examine the difference between the distributions in Figures 2 and 3 from tournaments using two configurations of the same card parameters.

Given playtesting input from actual players and historical data from other successful metagames, game designers can interpret the changes new cards have brought and implement changes to the game or metrics for optimization that will improve the experience of human players.

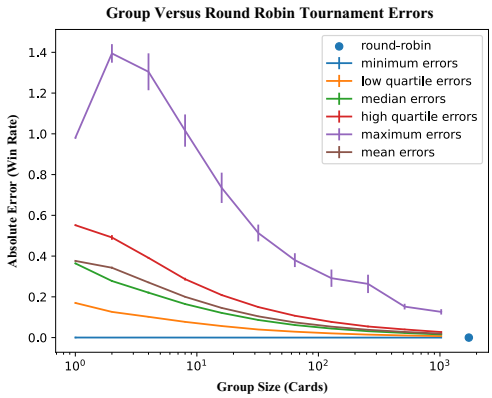

Figure 4: Absolute value of the difference in win rate between a *group tournament* of varying size and a full round-robin tournament of 1,728 possible lineups.

3.5 Experiment Design

Our experiments share common phases.

1. Build lineups. We generate a set of cards and their associated attack, health points, and special parameters. Depending on the experiment, a subset of these parameters may vary between simulations during the optimization process. From this set of cards, the lineups are permutations of 3 cards from the set, allowing for duplicates.
2. Run tournaments. This will be either be a group or a round-robin tournament as described in Section 3.1.
3. Optimize. These tournaments are run within each iteration of our genetic algorithm to produce a metric estimating the best possible health of the metagame with respect to configurations of the card parameters.

4 Results

We explore applications of LUDUS to problems that game designers commonly face.

4.1 Group Versus Round-Robin Tournament

The first experiment investigates the efficacy of our sampling-based approximation in Section 3.1. To do this, we compare the win rates of a complete round-robin tournament (every combination of lineups played) with the distribution of win rates over a 16-fold tournament of randomly partitioned lineups into groups of various sizes, playing every combination of the lineups within those groups.

Figure 4 compares the win rates of each lineup in the group tournament versus the round-robin tournament, and we can see that at a group size of 256, the mean error becomes negligible (about 0.0382). We use this group size in the other experiments rather than a round-robin of all 1,728 lineups.

4.2 Optimizing Cards without Special Mechanics

The next experiment optimizes a set of four 'vanilla' cards with no extra mechanics. We expected that the 'optimal'

solution in terms of the standard deviation metric would have four identical cards because the interchangeable cards should have zero standard deviation in the win rate. Due to the limited number of generations in our genetic algorithm, we recognized that there was no guarantee it would find this solution. From a game design perspective, a less-optimal solution seems more interesting in this case as a game with only a single card is obviously uninteresting. We wanted to see what 'almost-optimal' solutions appear for this setup. This observation also illustrates a case where the standard deviation metric fails to fully capture the qualities we wish to optimize in our game.

The results of this experiment illustrate a different failure case of our standard deviation metric that we did not predict. The genetic algorithm quickly found the solution (5/3), (5/4), (8/3), (8/1) where (a,b) indicates a card with a attack and b health points. This solution had the optimal zero standard deviation—why? Any pairing of these cards will result in both cards killing each other simultaneously. Thus, every game ends after one round of combat phases in a tie, and there is no variance in win rates amongst the lineups.

4.3 Optimizing Only Special Mechanics

The third experiment optimizes only the special mechanic parameters of twelve cards. We assigned each card an intuitive value for its attack and health, which a game designer might do for initial context. LUDUS then optimized the parameters for the special mechanics of these partially-designed cards. We were interested to see how it optimized under these constraints and how impactful the special mechanics would be for the overall balance. Table 1 describes the cards used in this experiment. The optimizer improved the win rate standard deviation from 0.0751 to 0.0589.

4.4 Optimizing All Parameters

After fixing the attack and health to focus solely on the special mechanics, the next experiment tunes the attack, health, and special mechanics all at once. To reduce the huge dimensionality of this experiment, we selected only five of the twelve cards to optimize. They are listed in Table 2.

The optimizer improved the win rate standard deviation from 0.0704 to 0.0584. This minimum is approximately the same as the one achieved in the previous experiment, modifying only the values of the special mechanics. One important observation we made is that the optimizer was still making good progress in the final generations of both our experiments. This indicates that these might not be minima, and compute time for additional generations could continue to improve the card designs.

4.5 Optimizing After a Set Rotation

Another common scenario designers of deck-building games encounter is releasing a new set of cards that maintain the compatibility, fairness, and competitiveness with the previously released cards. To apply LUDUS to this problem, we took the set of five cards from Section 4.4 with their optimized solution as the first set of cards released. Next, we selected five additional cards, described in table Table 3, and optimized them alongside the first set fixed at the solution found previously—this is ten cards with variable parameters for only the second set of cards.

The optimizer improved the win rate standard deviation from 0.0591 to 0.0541. This is a comparatively smaller improvement than the previous experiments, but the initial state was already in much better condition than any of the previous experiments. This is interesting because it could indicate that adding new cards to an already well-balanced set may not disturb the balance as much as one might expect.

5 Discussion

We can see that the LUDUS framework yielded some very useful results from a game design perspective. Given a small, yet representative, set of cards, our methods were able to find more balanced configurations for cards that should create a far more enjoyable auto battler game to play.

We also implemented an approximation method that considerably reduces the computation time necessary to evaluate configurations during optimization, especially for large card sets. This method randomly breaks the tournament into smaller groups that only compete within themselves. Our empirical results indicate that under most circumstances, the approximations are close enough to the complete tournament to produce satisfactory optimization results.

We applied the LUDUS framework to some standard problems game designers face and found promising results. Notably, the algorithm successfully optimized cards under various constraints from partially-designed cards to expansions of previously optimized card sets.

Beyond the immediate results of this paper, we contributed an auto battler game and its associated tools. Given the recent nature of this genre, we believe this is an important contribution that will aid future research in the area.

6 Future Work

The genre of auto battlers is very young, and our research only explores a narrow slice of questions in this new area. We discuss problems that extend our current work.

6.1 Other Auto Battler Features

Unlike our game, some auto battlers have players iteratively build their lineup between battles by purchasing cards from an array of options. While our experiments were concerned with fixed lineups of size 3, LUDUS's ability to quantify, qualify, and optimize balance for other deck-building rules remains to be assessed. We previously only considered the case where any card may be replaced with any other card to create a new lineup. However, the purchasing value of cards is another dimension of their design; future work should consider comparing lineups that players can build after the same number of battles and purchasing phases.

In many auto battlers, cards have multiple special mechanics that each have variable parameters. Our work only considers cards with a single mechanic, but this should be extended to optimize design for such cards.

Card	Attack	Health	Special Parameter (see Section 3.1)	Optimized Special parameter
Explode On Death	2	1	Explosion Damage	1
Friendly Vampire	1	3	Heal Amount	3
Grow On Damage	0	5	Attack Growth Per Hit	4
Heal On Death	1	2	Explosion Health	3
Health Donor	1	4	Heal Donation Percent	3
Ignore First Damage	2	1	Armor Points	1
Morph Attack	0	3	Morphing Enemies	N/A
Pain Splitter	2	2	Damage Split Percent	7
Rampage	0	4	Middle Age	8
Survivalist	2	2	Survivalist	N/A
Threshold	2	2	Target Age	5
Time Bomb	1	8	Detonation Time	7

Table 1: Variable and Fixed Parameters for the *Optimize Special Mechanics* Experiment

Card	Optimized Attack	Optimized Health	Optimized Special Mechanic (see Section 3.1)
Survivalist	9	1	N/A
Morph Attack	5	5	N/A
Ignore First Damage	6	8	1
Explode On Death	4	1	3
Vanilla	7	2	N/A

Table 2: List of Cards in the First Set and Their Optimized Solution

Card	Optimized Attack	Optimized Health	Optimized Special Mechanic (see Section 3.1)
Friendly Vampire	7	7	8
Grow On Damage	2	6	9
Heal On Death	3	9	1
Rampage	6	7	5
Vanilla	7	6	N/A

Table 3: List of Cards in the Second Set and Their Optimized Solution

6.2 Simulation Data

All metrics we present are based on the average win rates of lineups collected from the simulator. This does not take into account individual lineup-versus-lineup win rates, which are always 0 or 1 since our auto battler game is deterministic. One could represent these relations via a *domination graph* with lineups as nodes and a directed edge to the lineup that wins the head-to-head matchup (Goldman et al. 2021), game matrices with each lineup as a strategy, and more. Further investigation is required to determine whether perturbations of card parameters causing a degradation in metagame health correspond to any changes in these representations. There are also game design factors to consider besides win rates, such as game durations and intensity levels.

6.3 Alternative Optimization Methods

Our choice of a genetic algorithm for optimization is somewhat arbitrary. Genetic algorithms can conveniently handle the familiar integer values of card parameters, unlike other optimization methods. Gradient- and Hessian-based methods are difficult to apply to this problem due to our parameters not being differentiable. Had they been applicable, they would have solved an unmet need of validating the results.

6.4 Human Playtesting

Human playtesting is still required to determine the health of the metagame as we have not verified to what extent our metrics correspond to what humans deem as a healthy metagame. Designers with a historically healthy metagame can use those parameters to generate reference win rate distributions over lineups and cards using our LUDUS framework. These distributions can then shape new metrics and imply their ideal values for parity. Human playtesting feedback could also motivate other metrics based on the gameplay experience, such as the number of turns. While this work is an example of how designers can readily apply the LUDUS framework, researching human playtesting-validated results marks an important avenue of future work.

Acknowledgments

The authors would like the thank the anonymous reviewers for their helpful feedback, Nathan Ringo for providing computational resources to run the experiments, and everyone at SIFT for their support.

References

Bhatt, A.; Lee, S.; de Mesentier Silva, F.; Watson, C. W.; Togelius, J.; and Hoover, A. K. 2018. Exploring the Hearthstone Deck Space. In *Proceedings of the Thirteenth International Conference on the Foundations of Digital Games*, FDG '18, 1–10. Malmö, Sweden: Association for Computing Machinery. ISBN 9781450365710.

Blizzard Entertainment. 2019. Introducing Hearthstone Battlegrounds. https://playhearthstone.com/en-us/news/23156373. Retrieved September 4, 2021.

Chen, T.; and Guy, S. 2020. Chaos Cards: Creating Novel Digital Card Games through Grammatical Content Generation and Meta-Based Card Evaluation. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 16, 196–202. Worchester, Massachusetts, USA.

DeTora, M. 2017. Designing Hour of Devastation Cards to Meet FFL Goals. https://magic.wizards.com/en/articles/archive/play-design/designing-hour-devastation-cards-meet-ffl-goals-2017-07-07. Retrieved September 6, 2021.

Drodo Studio. 2019. Auto Chess–Official Website. https://ac.dragonest.com/en. Retrieved September 4, 2021.

Duke, I. 2019. November 18, 2019 Banned and Restricted Announcement. https://magic.wizards.com/en/articles/archive/news/november-18-2019-banned-and-restricted-announcement. Retrieved September 6, 2021.

Duke, I. 2020a. August 3, 2020 Banned and Restricted Announcement. https://magic.wizards.com/en/articles/archive/news/august-8-2020-banned-and-restricted-announcement. Retrieved September 6, 2021.

Duke, I. 2020b. June 1, 2020 Banned and Restricted Announcement. https://magic.wizards.com/en/articles/archive/news/june-1-2020-banned-and-restricted-announcement. Retrieved September 6, 2021.

Gad, A. F. 2021. PyGAD: An Intuitive Genetic Algorithm Python Library. arXiv:2106.06158.

Goldman, P.; Knutson, C. R.; Mahtab, R.; Maloney, J.; Mueller, J. B.; and Freedman, R. G. 2021. Evaluating Gin Rummy Hands Using Opponent Modeling and Myopic Meld Distance. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(17): 15510–15517.

Good Luck Games. 2021. Storybook Brawl. https://storybookbrawl.com/. Retrieved September 4, 2021.

Grayson, N. 2019. A Guide to Auto Chess, 2019's Most Popular New Game Genre. https://kotaku.com/a-guide-to-auto-chess-2019-s-most-popular-new-game-gen-1835820155. Retrieved September 4, 2021.

Jaffe, A.; Miller, A.; Andersen, E.; Liu, Y.-E.; Karlin, A.; and Popović, Z. 2012. Evaluating Competitive Game Balance with Restricted Play. In *Proceedings of the Eighth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, AIIDE'12, 26—-31. Stanford, California, USA: AAAI Press.

Kim, J. H.; and Wu, R. 2021. Leveraging Machine Learning for Game Development. https://ai.googleblog.com/2021/03/leveraging-machine-learning-for-game.html.

Konami. n.d. Yu-Gi-Oh! Trading Card Game. https://www.yugioh-card.com/en/. Retrieved September 4, 2021.

Nguyen, T.-H. D.; Chen, Z.; and Seif El-Nasr, M. 2015. Analytics-Based AI Techniques for a Better Gaming Experience. In Rabin, S., ed., *Game AI Pro 2: Collected Wisdom of Game AI Professionals*, chapter 39, 481–500. New York: A. K. Peters, Ltd. / CRC Press.

Nintendo Mobile. 2020. Fire Emblem Heroes - Loki Presents (Pawns of Loki) *[Video]*. YouTube. https://www.youtube.com/watch?v=Ldf3Np5lMhs.

Scott-Vargas, L. 2020. Luis Scott-Vargas on Twitter: "Is Throne of Eldraine really legal for another year? It feels like it's dominated Standard for years already..." / Twitter. https://twitter.com/lsv/status/1339644691378282496?lang=en. Retrieved September 6, 2021.

Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; van den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; Dieleman, S.; Grewe, D.; Nham, J.; Kalchbrenner, N.; Sutskever, I.; Lillicrap, T.; Leach, M.; Kavukcuoglu, K.; Graepel, T.; and Hassabis, D. 2016. Mastering the Game of Go with Deep Neural Networks and Tree Search. *Nature*, 529: 484–489.

Tack, D. 2019. What Is Dota Auto Chess And Why Is Everyone Playing It? https://www.gameinformer.com/2019/01/14/what-is-dota-auto-chess-and-why-is-everyone-playing-it. Retrieved September 8, 2021.

The Pokémon Company. n.d. Pokémon Trading Card Game | Pokemon.com. https://www.pokemon.com/us/pokemon-tcg/. Retrieved September 4, 2021.

Triske. 2019. [Magic: The Gathering] Come All Ye Christians and Learn of a Sinner: This Guy, Oko. https://www.reddit.com/r/HobbyDrama/comments/do8v2j/magic_the_gathering_come_all_ye_christians_and/. Retrieved September 4, 2021.

Wizards of the Coast. n.d.a. Banned and Restricted Lists. https://magic.wizards.com/en/game-info/gameplay/rules-and-formats/banned-restricted. Retrieved September 6, 2021.

Wizards of the Coast. n.d.b. Magic: The Gathering | Official Site for MTG News, Sets, and Events. https://magic.wizards.com/en. Retrieved September 4, 2021.

Xu, J.; Chen, S.; Zhang, L.; and Wang, J. 2020. Lineup Mining and Balance Analysis of Auto Battler. In *Proceedings of the IEEE Sixth International Conference on Computer and Communications*, 2300–2307. Chengdu, China.

Zook, A. 2019. Better Games through Data: How Blizzard Uses Data Science for Epic Experiences. Recording available at https://youtu.be/_YSYVRdzUkE?t=2636. Last accessed September 4, 2021.

Zook, A.; Harrison, B.; and Riedl, M. O. 2019. Monte-Carlo Tree Search for Simulation-based Strategy Analysis. *CoRR*, abs/1908.01423: 1–9.