# Learning How Design Choices Impact Gameplay Behavior

anon, *Member, IEEE*

***Abstract*—Designers structure a game to provide a desired range of player behaviors: the *play space* of a game. Any given game is one instance from a *design space* of alternatives. Navigating a design space to achieve a designer's goals requires knowledge of how design choices shape the play space in a game. We present algorithms to automatically measure play patterns using statistical models that predict how design choices alter player behavior. We present Monte-Carlo Tree Search as a way to sample behaviors from a play space, *action metrics* to automate play space measurement, and predictive modeling techniques to model design spaces. We demonstrate these techniques in two simplified, perfect information, adversarial game domains based on *Scrabble* and *Hearthstone* showing their use for automated design space modeling.**

***Index Terms*—Game design, Procedural content generation**

## I. Introduction

GAME designers create games to give players a desired interactive experience. Crafting interactive experiences is challenging: designers intend for players to behave in certain ways, but are only able to make choices about the mechanics and content in a game [1]. Designers typically address this challenge through a process of observing people play a game and iteratively adjusting the game's design to produce desired player behaviors [2], [3]. Through this process designers see what behaviors a game affords as well as how changes to the design shape that behavior in different ways. Conceptually, this decomposes into seeing about the range of possible behaviors within a design—the *play space*—and seeing how design variants shape that behavior—the *design space*.

The process of exploring and modeling the space of play and design space is challenging, but essential to successful game design. How can machines enhance this process? Machine assistance can benefit human designers by: automating mundane aspects of design, improving the efficiency of varying a design, uncovering strengths and flaws in a design, or even providing knowledge about how a design works. Ultimately, these techniques may even enable machines to automatically develop and refine games on their own or leverage the scale of computation to explore broad swaths of games that humans cannot readily conceptualize. Enabling these applications requires us to operationalize the process of measuring a space of play and modeling a design space. Here we define the *space of play* of a game as the range of behaviors players take in a game, represented extensionally by a set of example sequences of behavior (termed playtraces). We define the *design space* of a game as the range of possible variants on the configuration of game elements, represented by parameterized game elements.

anon are in the a School of at a University.
Manuscript received March 20, 2017.

Computationally measuring a space of play requires (1) a source of example behaviors (called playtraces) and (2) ways to aggregate example behaviors into summarizing metrics. We term the problem of producing examples of player behavior in a given game the *behavior sampling problem*. Ideally a behavior sampling algorithm should apply to many game designs, allowing its reuse across domains. As players learn to play (most) games over time, we can expect player capabilities to vary and thus the algorithm should also model player capabilities. For this work we studied two simplified versions of existing two-player, adversarial, turn-based games, made into perfect information games: one based on the word game *Scrabble* and another based on the card game *Hearthstone*. While these games exercise many capabilities, we specifically focused on modeling the ability to plan ahead. We use Monte-Carlo Tree Search (MCTS—a stochastic planning algorithm) as a general behavior sampling algorithm, adjusting the resources used by the algorithm to model differences in player planning capabilities.

Once example behaviors are gathered we require a method for *gameplay analysis*: summarizing the instances of behavior from a space of play into aggregate metrics. But what aspects of play should be aggregated? We contend that the *actions* taken in a game are a key part of understanding the space of play in a game. This contrasts with traditional game analysis metrics that primarily concern the *content* players reach in a game—for example, the levels of the game completed or enemies defeated. We argue player capabilities are often best represented directly by the choices made by players. For example, the strategic choices made to beat a boss in a *Zelda* game can be more informative for changing the game than knowing how much damage was inflicted to the bosses in the game. Further, in competitive games the actions taken by players are the core element of design used to balance the game. We present four levels of granularity for aggregating player actions into metrics, and use these to analyze our case study games to reveal their design strengths and flaws.

With the ability to measure a play space (summarized by *action metrics*) a machine can compare alternative game designs. This enables the systematic evaluation of a wide space of game designs. Generating and evaluating many design variants of a game enables a machine to build models of how changes to a game alter the space of play. Searching this design space allows a machine to optimize a game's design toward desired metrics on the space of play. Further, comparing variants of a game's design enables the machine to build predictive models of expected metrics for the play space of a game based on the choice of game design configuration. To this end, we show an approach to optimize the design

of an adversarial card game, and predict how card design configurations alter its play space.

In this work we make four contributions toward enabling machines to acquire game design knowledge:

1) Present Monte-Carlo Tree Search as a general behavior sampling algorithm to simulate player behavior and model differences in player planning capabilities.
2) Define four types of metrics to analyze the actions in a space of play.
3) Demonstrate automatically measuring the space of play in two turn-based, adversarial games.
4) Demonstrate building predictive models of how design choices alter the play space in our *Hearthstone*-like game.

Below we first review related work on the problems of behavior sampling, gameplay analysis, and learning design knowledge. We then present the simplified versions of *Scrabble* and *Hearthstone* we studied. We report on two experiments on these games: (1) analysis of the play space of these two games and (2) learning about the design space of the *Hearthstone*-like game. We conclude by discussing limitations of the current work and future research directions.

## II. Related Work

Here we review related work on behavior sampling, gameplay analysis, and learning design knowledge.

### A. Behavior Sampling

There are three primary ways to solve behavior sampling: (1) human playtesting, (2) model-checking, and (3) player simulation. Each approach provides different information about a game's play space. Playtesting gathers observations of people playing a game, ensuring that the gathered information represents how humans interact with a game [3], [4]. Playtesting has two primary limitations: (1) it may not be representative and (2) it is not exhaustive. First, playtesters may not act like the final player population and it can be difficult or impossible to foresee this difference. Second, humans rarely test all ways to play a game to probe the bounds of the play space.

Model checking methods offer a way to exhaustively explore what actions are possible in a game [5]–[9]. Model checking methods can determine what is (im)possible in a game, but come with two limitations: (1) representing the likelihood of behavior and (2) scaling to complex games. First, checking a model entails (intelligently) exploring all ways a game can unfold, but this process does not readily provide information on which ways of playing are more or less likely. Second, exploring all game playtraces means model checking cannot readily scale to games with large state spaces or complex mechanics. Large games create an explosion of states to explore, which can only partially be mitigated by abstracted representations of game states. Games with stochastic events are also difficult to handle using standard deterministic model checking methods.

Simulation methods construct a computational agent to play a game instead of a human player. Unlike model checking, simulated agents can model which actions players are likely to take and do not need to exhaustively explore a play space. This comes with the obvious limitations that agents may not accurately reflect all players (as in human playtesting), but with the advantage of being faster and cheaper than humans. Agents can be trained to reproduce many aspects of human-like play including reaction time [10], memory and planning [11], movement behavior [12], and action choices in many genres [13]–[16]. Simulated players offer two advantages for our work: (1) controllable variability and (2) balancing exploration and exploitation of the play space. First, simulated agents readily incorporate parameters to vary their computational resources, enabling them to model how highly skilled players may pursue entirely different strategies to novices [17]. Second, agents can use randomized techniques and take probabilistic expectations to search the most important parts of the space of play, balancing exploring different playstyles with providing more information on expected playstyles.

In this work we use a stochastic planning technique with demonstrated success in general game playing: Monte Carlo Tree Search (MCTS) [18]. Unlike prior work on MCTS playing games to win, we use MCTS as a tool to sample the space of play. Varying the MCTS agent's computational resources affords us a proxy for varying player skills, capturing how the space of play differs for different players.

### B. Gameplay Analysis

Gameplay analysis aggregates playtraces from a play space into metrics summarizing that space. Playtraces may be viewed as a sequence of states or a sequence of actions, providing complementary information on *where* players go in a game (states) and *what* players do in a game (actions). State analysis summarizes the content players consume and how they progress through a game [4], [19], [20]. By contrast, action analysis summarizes the choices players make and the strategies and mechanics players do (not) use [17]. We contend that action analysis helps understand the play space of games in general and adversarial games in specific: strategic choices reflect important parts of the decision-making process in games. As such, we present *action metrics* at four levels of granularity for evaluating the actions in a game's play space, enabling analysis of the strategic space of games.

### C. Design Knowledge

Design knowledge comes in many forms: recognizing patterns in the content in games, summarizing and predicting patterns of player behavior, or patterns for how design features produce patterns of player behavior. Knowledge of *patterns of game content* is learned by abstracting structures in game content across many instances in a given game, with early work using human analysis to compile these patterns [21]. Computationally modeling game level structures from corpora of levels enables systems to generate, repair, and critique existing levels and can allow designers to explore a space of level designs [22], [23]. Knowledge of *patterns of player behavior* (often studied in player modeling [24]) is learned by observing player behavior in games to predict expected player actions or subjective ratings of different game content.

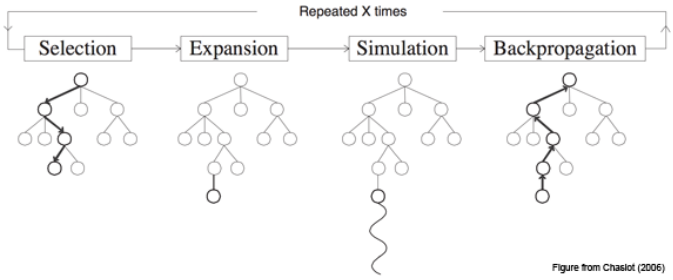


Fig. 1. MCTS algorithm steps [31]

For example, player models for platformer games enable systems to generate levels tailored to player preferences [25] or playstyles [26]. Unlike knowledge of content structure or player behavior, we use the term *design knowledge* for patterns of how design changes alter player behavior.[1] Design knowledge is learned by abstracting across instances of game designs and their accompanying player behaviors to model how changes to a design will change player behavior.

Design knowledge enables systems to predict how changes to a game will alter player behavior without requiring the system to directly observe a game with that design. This allows the system to gather general, reusable knowledge that can serve many purposes and contexts, rather than producing a single, optimal design for a pre-defined criteria as in search-based procedural content generation [28], [29]. In this work we demonstrate learning design knowledge in a two-player, turn-based, adversarial card game. We generate a wide space of design variants, sample from and summarize the play space in each, and learn predictive models for how design features change action metrics summarizing these games. We use this design knowledge to generate games that optimize specific design qualities and to provide models to people that describe the design space of this card game domain.[2]

## III. Behavior Sampling

Here we describe our general approach to behavior sampling using Monte-Carlo Tree Search and the two game domains used in the subsequent studies. Both domains were simplified to be perfect information; we leave games of hidden information to future work.

### A. MCTS Agents

Monte-Carlo Tree Search (MCTS) is a general game-playing technique with broad success in discrete, turn-based, and non-deterministic game domains [32]. MCTS is a sampling-based anytime planning method that can use additional computational resources to more fully explore a space of possibilities, allowing control over the balance between computational time and exploration of the full play space

[1]Work in both content and player modeling sometimes matches this definition (see [26], [27]), though these efforts typically aim to understand players or generate content, rather than analyzing design choices.

[2]Our optimization approach conceptually aligns with methods that approach content generation through the lens of exhaustive generation followed by careful selection, though we approximate exhaustive generation through sampling due to our need to generate gameplay behaviors [30].


Fig. 2. A digital recreation of the word game *Scrabble*.

(Figure 1). We chose MCTS as a behavior sampling algorithm for it's proven high-level performance, domain generality, and variable computational bounds. For simplicity, we altered the study domains to be perfect information (allowing players to see one another's hands) to facilitate use of MCTS. We use the UCB1 algorithm [33] to ensure the agents fully consider options for a given move before planning further, later moves.

For an even comparison across game domains we used MCTS agents to play both study games, altering the number of rollouts used as a proxy for human ability to plan ahead [11]. *Rollouts* are the number of times the algorithm considers move choices. An agent with more rollouts can more fully explore the value of possible actions in the game and better plan future actions (Chapter 5, p. 60 in [34]).

Considering differences based on player planning ability enables analysis of many design questions: Does the game reward stronger players with greater rewards or higher win rates? Do players with different skill have different play styles? Does the game enable a smooth progression of skill as players learn over time? In adversarial games, varying the rollouts used by two MCTS agents can compare how gameplay looks when two agents having varying levels of skill, as well as compare the effects of relative differences in skill between two agents; e.g., comparing high-level play between two strong agents or comparing games between a weak and strong agent. This improves over human testing as it affords designers the ability to readily compare games between players with many skill combinations.

### B. Scrabble

*Scrabble*[3] is an adversarial game where players take turns placing tiles onto a game board to create words (Figure 2). We implemented the standard version of *Scrabble*, which includes bonus spaces to double or triple letter tile values on the board. Normally games end only when one player runs out of tiles,

[3]The *Scrabble* domain and analysis was done by Brent Harrison in joint work on applying MCTS to behavior sampling [35].

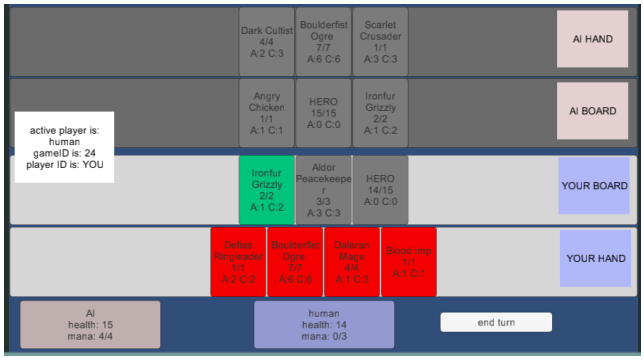


Fig. 3. *Cardonomicon*, a minion-based card game.

with the winner as the player with a higher score. We changed the game to end with the winner as the first player to reach 150 points. This simplification improves search performance in the domain and does not adversely affect the ability to ask design questions, as seen in the study evaluations.

Moves in *Scrabble* are often represented as placing tiles on the board. This representation, however, requires the MCTS agent (or game) to validate whether the tiles form legal words. Instead, in our implementation the agent represents moves on a turn as the word formed on that turn, using a dictionary to choose valid words. Thus, the space of possible moves on a given turn is all possible words that can be made on that turn.

### C. Cardonomicon

We built *Cardonomicon* using the core elements of adversarial card games like *Magic: The Gathering* and *Hearthstone* (Figure 3). From a design perspective, *Magic* and *Hearthstone* are difficult to balance due to the difficulty of predicting the strategies players will develop to play the game. The design is highly sensitive to interactions among mechanics: each card must be balanced with respect to all other available cards; e.g., a single overly powerful card can make all other cards irrelevant. Further, the random order of card draws and non-deterministic effects of actions introduce a large space of non-deterministic outcomes to sample over. While *Magic* and *Heartstone* have hidden information, for simplicity *Cardonomicon* is perfect information. In addition, we fix the decks used by players, rather than allowing deck construction—this drastically reduces the search space for behavior sampling while preserving properties that make this a domain of interest.

In *Cardonomicon*, two players start with an identical deck of 20 cards representing minion creatures (see Appendix A, Table V for card definitions). Gameplay consists of drawing cards, spending mana to place cards on the game board, and using cards to attack one another and the opposing player's hero. Cards are parameterized by health, attack power, and mana cost. Players start with a single hero card on the board with 20 health and 0 attack; a player loses when their hero's health is reduced to or below 0. Each turn, players may play any combination of cards for which they can pay the mana costs. A player's mana starts from 1 on the player's first turn and increases by 1 each turn up to a cap of 10. Playing cards puts them on the board, making them available to attack any opposing card once per turn. When a card attacks, the opposing card's health is reduced by the attacker's attack; attacking cards receive counter damage. We designed a set of cards to allow the player to play one of multiple cards on each turn (with differing parameterizations), assuming they have drawn a playable card.

In *Cardonomicon* the MCTS agent represents possible moves as either playing a card or using a card to attack another card on the opponent's board. One turn may involve multiple moves in a row. The agent has one move for every card that can be played in the agent's hand and one move for every pair of their card attacking a target opponent card. Only cards that may attack are represented and no attacks on the agent's own cards are permitted as this has no purpose in the *Cardonomicon* domain. One additional move to end the turn is always available. Thus, MCTS agents reason at each turn about whether to play a card, use a card to attack the opponent, or end their turn.

## IV. Measuring a Space of Play

In this section we present metrics at four levels of granularity to summarize a play space in terms of player action choices. We apply these metrics to *Scrabble* and *Cardonomicon* to show how these metrics can reveal design strengths and flaws in the two domains, respectively. The two case studies demonstrate measuring the strategic play in a game and using this in human design analysis.

### A. Action Metrics

Decision-making in games is based on many contexts of varying levels of abstraction. These contexts span the utility of an action in itself, to responding to other player actions, to considering the long-term implications of a choice later in a game. We divide analyses of player actions into *action metrics* at four levels of abstraction:

- *Summaries* are high-level design metrics that aggregate playtrace features of interest. For example, the typical (median) length of the game or probability of the first-turn player winning in *Scrabble*.
- *Atoms* are metrics specific to individual actions in a game. For example, the frequency of playing a letter in *Scrabble*, potentially conditioned on a context like the turn number in the game.
- *Chains* are gameplay patterns within or between players. *Combos* are regularities in actions taken by a single player: e.g., in *Magic*, tending to play a given pair of cards on the same turn (potentially due to positive synergies between the cards). *Counters* are action-reaction patterns in actions taken between a pair of players: e.g., in *Scrabble*, when one player spells "con" the opponent may often add "i" to form "icon."[4]
- *Action spaces* are sets of actions taken (or available) to a player, potentially over the course of a game. For example, in *Scrabble*, the number of valid words available

[4]Our definitions for 'atom' and 'chain' are distinct to those proposed by Dan Cook [36], but share the notion of distinguishing between single actions as atoms and patterned sequences of actions as chains.

to be played over the turns of a game or in *Magic*, the number of unique minions a player can play on each turn.

These categories are not intended to encompass all ways to analyze a play space, but instead organize levels of analysis that share common techniques in terms of aggregating descriptive statistics and visualizing those results. These metrics only require sets of play traces as input and can equally apply to traces from humans or simulated agents. By only referencing actions taken in a game all of these metrics can be subdivided by features of game players: here we consider player skill, though other features may be of interest (e.g., player gender or age [37]). Action metrics allow designers to survey common patterns of play in a game. The metrics on their own do not enable a designer to infer the causes for these patterns or understand the detailed course of action that yields the metrics. These and other aspects of inference for a design require alternative analytic techniques outside the scope of this work. Below we clarify these definitions and give examples for *Scrabble* and *Cardonomicon*.

*1) Summaries:* Summaries overview gameplay features to provide high-level information to guide further analysis and framing to interpret more granular analyses. Summaries are typically single numbers that aggregate features of a game: average game length, number of users of the game, revenue earned per (paying) user, etc. Example summaries in *Scrabble* and *Cardonomicon* include: typical game length, probability of the first-turn (or more skilled) player winning, and number of actions taken during the game. Other summaries include: game play duration, typical turn duration over the course of the game, number of actions taken in turns in a game (overall and split over the course of the game), probability of winning for players of different skill levels, average time alive during a match in multiplayer games, total narrative choices made, average length of narrative story paths followed, and so on.

*2) Atoms:* Atoms summarize the use of individual actions in a game, providing information on which game mechanics are (not) being used and are (not) available to be used. Analyzing atoms can inform game balancing decisions around whether specific actions are over- or under-used in the game. Actions come in many forms: e.g., words to make in *Scrabble* or cards to play in *Cardonomicon*. Other examples include: plot choice frequency in interactive fiction, ability use rates in a role-playing game, item use or purchase rates in games with inventories, avatar/character choice rates in games with multiple avatars. Action analysis can consider both the actions *taken* by agents as well as the actions *available* to agents to use. Understanding action use provides information on how readily actions can be used and whether those actions are (perceived to be) effective in the game. Comparing action choices made by players with different capabilities can demonstrate to what extent a game can be learned by agents, where a lack of ability to learn to play may be a design flaw.

*3) Chains:* Chains summarize recurrent sequences of actions in playtraces. *Combos* are sequences of actions a single player commonly uses together. Combos are common in games with multiple actions per player turn or real-time action. In *Cardonomicon* combos include playing cards successively or using sets of cards to attack; *Scrabble* has no combos as players take a single move each turn. Other combos include: attack sequences in a fighting game, sets of equipment used together, cards placed in a deck, monsters or party members associated with a team, path snippets followed on a multiplayer competitive game map, or build orders in a strategy game. *Counters* are sequences of actions that occur when two (or more) players respond in similar ways to actions from other players. Counters are common in games with alternating turns or simultaneous turns. In *Cardonomicon* counters can occur when one player plays a card on the board and their opponent attacks it using a specific other card; in *Scrabble* counters occur when one player forms a word and their opponent builds a longer word from that base. Other counters include: units deployed to counter an opponent in a strategy game, spell attacks chosen against buffs in role-playing games, attack/block/throw choices against an opponent attack in fighting games, or positional choices in chess against opponent moves. Analyzing chains can reveal emergent strategy within a game, including chains of actions that may exercise a skill [36] or ways players have discovered to thwart their opponents [38]. Understanding which combos or counters are common can inform decisions to alter the restrictions placed on using an action or alterations to how effective an action is. Segmenting analysis of chains by player skill can reveal how player strategies evolve with greater proficiency in the game and reveal balance concerns if specific actions disappear from chains used in high-level play.

*4) Action Spaces:* Action spaces summarize atom use over time or game states. In *Scrabble*, action spaces include the number of distinct tiles played or the number of distinct words available to complete across turns in a game. In *Cardonomicon*, action spaces include the number of distinct cards available to play or average number of cards able to attack across turns. Other action spaces include: frequency of item purchase by location or vendor in a game, location visits in a map of a tactical or strategy or platformer game, ability use rates over the course of a MOBA game, or unit build rates or economy income over the duration of a strategy game. Analyzing actions spaces can reveal how a game progresses from the perspective of player choices. This analysis can identify cases where a game is too restrictive or overwhelming with too many options, informing decisions about the pacing and growth of game complexity over time [17]. Considering differences in actions spaces between low- and high-skill players can reveal cases where skill allows better use of the game actions or where low-skill players fail to use actions commonly used by high-skill players.

### B. Experiment Design

Our studies use MCTS agent pairs of varying computational bounds as a proxy for varying player skill to handle behavior sampling. We varied agent reasoning to consider roughly one to two moves ahead in the game. Two moves ahead is an upper bound potentially relevant to human play; research in reasoning on recursive structures suggests people are able to reason to roughly two levels of embedding. Models of deductive reasoning on logic puzzles support this claim [11]. The MCTS selection policy (UCB1) we used forces trying

all moves available after making a given move once before repeating a move: thus all rollouts will first explore options for a single move before exploring two-move sequences. Random moves are chosen during rollouts.

To set computational bounds we approximated the average number of choices available to an agent over the entire game and used this number to estimate the number of rollouts an agent would need to consider one or two moves ahead in the game. For equity we modeled agents with fixed computational resources in a turn, meaning they consider an equal number of choices per turn, regardless of the number of potential available choices on that turn. To examine a range of agent capabilities we initially created three agent computational bounds (number of rollouts allowed):

- A *weak* agent able to explore all moves on a given turn, but lacking resources to explore to two moves ahead.
- A *strong* agent able to fully explore moves on the current and next (opponent's) turns. This allows modeling the impact of opponent action choices on the focal agent.
- A *moderate* agent with rollouts halfway between these two.

Initial testing revealed little difference between the latter two agents, possibly due to marginal returns for greater computational resources in our study domains due to their large branching factor. We thus halved the number of rollouts of the two stronger agents, making the moderate agent weaker than before, and thus having less benefit from the amount of search it could perform. This created a spectrum of agent capabilities that clearly illustrates the influence of differences in player skill.

For each game domain we ran a pair of agents where each agent was set at one of the three levels. We simulated 100 games for each pair to get aggregate statistics on agent performance. In *Scrabble*, we approximated the number of rollouts for a single level deep by looking at the median number of possible words an agent could complete on the board in a single turn, summarizing over all turns in the game: 50. Thus, the weak agent used 50 moves. Initially the strong agent was allowed 2500 rollouts ($50^2$ for two moves ahead) and the moderate agent 1250 rollouts. After halving, this resulted in a moderate agent with 650 rollouts ($(1250-50)/2+50=650$) and a strong agent with 1250 rollouts. In *Cardonomicon*, we approximated the number of choices of playing cards as choosing 2 cards to play each move out of a hand of 6 cards ($\binom{6}{2}=15$ moves). We modeled attack choices assuming the player (and opponent) have approximately 3 cards on the board and one hero card, yielding 3 source card choices for 4 targets ($3^4=81$ moves) Together this yields a total of approximately 100 moves considered for the weak agent, 10000 for the strong, and 5000 for the moderate. After halving this resulted in 100, 2500, and 5000 rollouts for the weak, moderate, and strong agents, respectively.

Note that an alternative strategy to sampling up to two levels deep would be to have agents explicitly model a selection policy with pure exploration up to one or two levels. In this case, search bounds would vary over the course of the game. We chose to use a fixed number of rollouts to capture the notion of agents of fixed 'capability' in terms of resources to devote to the problem.

### C. Results

For the two game domain cases we examined the four skill-based design metrics above: summaries, atoms, chains, and action spaces. In the *Scrabble* domain these metrics highlight how the game is balanced and illustrate how player skill differences manifest as differences in skill-based metrics. In the *Cardonomicon* domain these metrics reveal imbalances in the design of the simplified game. Together, these studies illustrate that skill-based design metrics can inform designers about the strategic space of play in a game.

*1) Scrabble Metrics:* The *Scrabble* domain shows how skill-based metrics reveal balance and player skill differences despite changing the game to end at 150 points. The study shows these changes did not upset the game balance and demonstrate that *Scrabble* rewards high skill play.

*a) Summaries:* For *Scrabble* we examined summaries of win rates and the length of a game in turns. We expected stronger players will consistently defeat lower-skilled opponents; however, it is unclear how skill would affect game length. Stronger agents consistently beat weaker agents, with a 91% win rate for a strong vs weak agent and 64% win rate for a strong vs moderate agent. First turn players, however have no clear advantage, with 58%, 55% or 49% win rates for evenly matched weak, moderate, or strong agents. Thus, *Scrabble* is balanced such that agents can exhibit greater skill, but do not have an inherent first turn advantage.

Strong agents have shorter games: games last 26 turns on average for evenly matched weak agents and 22 turns for strong agents. This is likely because skilled opponents make moves worth more points, reaching the 150 point ending criteria sooner: stronger agents typically play longer words, corroborating this conclusion (see below). An alternative explanation for the shorter games is that stronger agents make better use of bonus tiles on the board, increasing individual word scores, and reaching the 150 point end sooner. However, agents of different strengths did not differ in their use of bonus tiles. Thus, the main cause of score differences between agents was the length of words played.

*b) Atoms:* In *Scrabble* we studied the atom of word use, finding weaker agents play shorter words while stronger agents play a wider variety of words (Figure 4). This provides further evidence of variable expressions of agent skill in *Scrabble*.

*c) Chains:* In *Scrabble counters* are the words played by the opponent after a word has been played by the other player. We used frequent itemset mining on the words played in two turns in a row to produce common counters. Among the top itemsets of words created across two turns, most counters either add to the previously played word, or build a two or three-letter word off of the word that was previously played. For example, one of the top counters to a player playing the word "con" on a turn was to add an "i" to the beginning of it to make the word "icon." This is not unexpected as building off words that were previously played will typically result in a higher point total since the player is playing a longer word than the opponent.

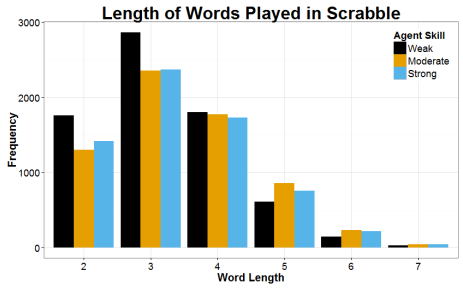


Fig. 4. Word length frequency in *Scrabble* by skill.

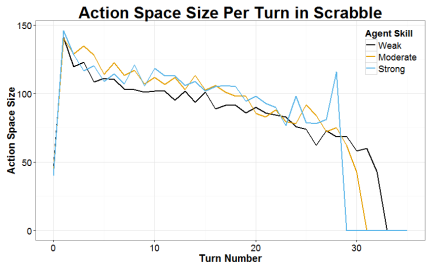


Fig. 5. Median number of words that could be played per turn based on skill.

*d) Action Spaces:* In *Scrabble* the action space can be characterized by the number of possible words that can be played and were actually played. Figure 5 shows the median number of *possible* words that could have been played on a given turn based on skill. The space of possible actions shrinks over the course of the game, likely because valid word placements become fewer later in the game. The figure also shows that stronger agents have more possible actions on a given turn than weaker agents.

Figure 6 shows use of the action space over the game in terms of distinct words played across all games. The space of words played shrinks faster for stronger agents than weaker agents, likely because stronger agents identify moves worth more points and avoid the rest of the action space.

*2) Cardonomicon Metrics:* The *Cardonomicon* domain shows how action metrics can identify design flaws. Recall that *Cardonomicon* is highly constrained in terms of the types of cards that are available to use and the types of decks that players can use. These major alterations we introduced to the typical structure of a card game negatively impacted the balance of the game.

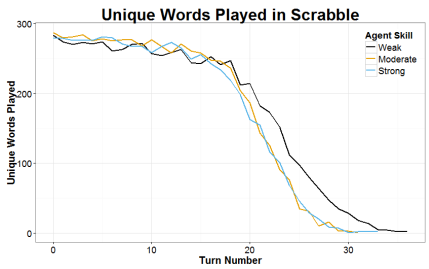


Fig. 6. Number of unique words played per turn based on skill.

*a) Summaries:* A core design flaw in *Cardonomicon* is the player going second has a large win rate disadvantage: a strong agent only has a 41% win rate against a weak agent, with 23%, 32%, and 32% win rates for evenly matched weak, moderate, or strong agents. While agent skill influences player win rates in *Cardonomicon*, the game gives a strong disadvantage to the player taking the second turn. This is expected due to the simplification of mechanics from *Hearthstone*: in *Cardonomicon* cards are able to attack and receive damage in retaliation, but the second player has no advantage in being able to play more cards on their first turn. As such, the second player will always deploy cards after the first player, but lacks a mechanism to catch up to the player who acts first.

Stronger agents have (slightly) longer games when matched to evenly skilled opponents: median 16, 17, and 18 turns for the weak, moderate, and strong agents, respectively. We attribute this trend to stronger agents being able to better counter one another while retaining enough cards to play until the end of the game.

*b) Atoms:* In *Cardonomicon* we studied the atoms of playing cards or using cards to attack. Stronger agents play more cards, but show no large differences in their use of specific cards. This likely indicates the deck size in *Cardonomicon* is too small: agents will play all of their available cards faster than they draw new cards and thus have no opportunities to favor playing specific cards against others. Stronger agents also use cards to attack more. Three cards showed disproportionately greater use by stronger agents compared to weaker agents: these three cards all had large amounts of health but low attack for their cost. Strong agents use these cards to destroy multiple weaker cards by intelligently trading off card attacks and retaliations. That is, stronger agents recognized the value in using a card with low attack (but high health) to remove several cards with lower attack and health over multiple turns. This confirms *Cardonomicon* allows for a limited form of strategic variety and supports the notion that MCTS rollouts can help detect these potential strategic variants dependent on player skill.

*c) Chains:* In *Cardonomicon* chains are primarily combos: sequences of actions taken by a single player in a turn of the game. As expected from the atom analysis, there were no significant combos in terms of playing or attacking cards. This is likely due to the lack of any strong synergy among cards in *Cardonomicon*: no pairs were particularly outstanding as no pairs have synergistic effects. This highlights another way to detect design flaws through these metrics: the absence of chains indicates no strong synergies exist in the design for players to use in combos.

*d) Action Spaces:* Stronger *Cardonomicon* agents have a larger space of cards they may play (not shown) and use to attack (Figure 7). Specifically, stronger agents have more options to play cards late in the game, while having fewer mid-game attack options with more late-game attack options. These results match intuition: in the early game both weak and strong players have a similar range of options constrained primarily by the amount of mana players have. By mid-game stronger players will have fewer attack options as they retain cards they may play for the late game. Playing these cards in

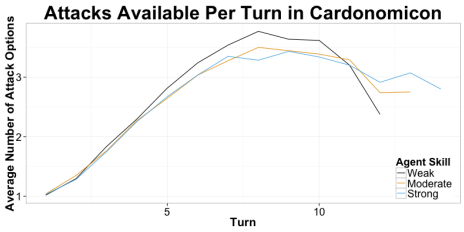


Fig. 7. Average number of possible attacks per turn based on skill.

the late game leads to more options to attack. Aligning with these analyses of the number of *possible* plays, more skilled players both play and attack with a larger number of cards on average. Thus, skilled players also actually use this larger set of options. Overall, these results demonstrate that more skilled players in *Cardonomicon* will open more plays in the mid-game by intelligently retaining cards before using these cards in the late game; in sum, these players are more efficient in their use of mana.

## V. Modeling a Design Space

In this section we first introduce predictive models of how changes to a design will change the play space of the design. We then apply this to *Cardonomicon*, showing how a system can predict changes to game length, playing cards to the board, and attacking with cards based on changes to a card's design.

### A. Predictive Design Models

Here we characterize design spaces using predictive statistical models that relate choices of design parameters to metrics summarizing the play space of the game. For example, the statistical model would indicate that card health is positively correlated with card use rates. Predictive modeling offers a number of benefits: informing choices of design iteration, aggregating play patterns shared by design variants, and exposing unexplored design opportunities. A designer (human or machine) can use these models to pick a design change to best accomplish their design goals. By building these models for many design variants a machine can provide an overview of the design space. For humans this aggregation can be valuable when it is difficult or impossible to track all the effects a design had on player behavior over a long period of game development. For machines this accumulation of knowledge contrasts with other approaches to computational design where an algorithm iteratively optimizes a single design, discarding any learned information after a design is found to optimize the (current) objective [28], [29]. By accumulating this information a machine can also expose what aspects of a design are poorly explored (missing input parameters) and what aspects of the space of play are poorly understood (highly uncertain).

In the following sections we discuss two applications of predictive design modeling: (1) optimizing a design for an arbitrary outcome and (2) modeling how card parameters alter the space of play. The optimization case highlights how predictive models allow manual exploration of how design choices alter aspects of the space of play. The modeling case highlights how design knowledge can be summarized into interpretable models that reflect human design intuition.

TABLE I
Designs optimizing desired play space features.

| | card configuration | | | optimal |
|---|---|---|---|---|
| **design goal** | health | cost | attack | value |
| least turns | 7 | 1 | 7 | 15 |
| most turns | 7 | 4 | 1 | 17 |
| most attacks | 7 | 1 | 1 | 1.44 |
| most attack options | 7 | 1 | 1 | 18.56 |
| most plays | 4 | 1 | 1 | 0.57 |
| most play options | 4 | 4 | 7 | 4.01 |

### B. Experiment Design

For our experiment we altered a single *Cardonomicon* card—"Stonetusk Boar", which defaulted to 1 attack, 1 health, and 1 cost—to model how a minimal design change shapes the play space in the game. Card variants differed in attack, health, and cost parameters, taking values of 1, 4, or 7 for each parameter. These variants cover the range of reasonable card configurations, yielding 3 attack values $\times$ 3 health values $\times$ 3 cost values $=$ 27 design variants.

Behavior sampling of each design variant paired agents of differing strength with balanced first turn assignment. Agent combinations covered all agent pairings with different strength: weak vs moderate, weak vs strong, and moderate vs strong. We ran 100 simulations for each for each card variant, agent configuration, and first turn agent combination to account for non-deterministic game mechanics. Together this required: 27 card configurations $\times$ 3 agent configurations $\times$ 2 first turn players $\times$ 100 simulations $=$ 16200 playouts. Below we analyze the play spaces of these design variants for two applications: (1) finding an optimal game design in this design space and (2) learning design knowledge of how design parameters shape player behavior.

### C. Design Optimization Results

Design optimization entails finding a game configuration with an optimal value for a desired play behavior. Having evaluated the play spaces of many design variants, finding an optimal design in the design space is simply a database query for the design with the optimal play space feature(s) [30]. To account for random variations in gameplay we averaged play space features across all playouts for a given card configuration. Using this approach we found designs that optimize games for a number of features of design interest, reporting the optimal value possible for the design goal and the features that achieve that value (Table I).

As examples, we found the designs and optimal values for the summary metric of game length and the atom metrics of playing or attacking with the focal card ("Stonetusk Boar"). A single card can have a modest effect on the game, adding 2 turns, or roughly 13% to the game's length. As expected, giving the card lower attack and higher health prolongs a game, while higher attack reduces game length, particularly when the card has high health and thus remains a threat for

longer. Low attack and high health allows more opportunities to attack with a card; low cost allows a card to be played more often and also attack. Together these results show how to explore a design space with play space metrics.

### D. Design Modeling Results

We model how choices of card parameters alter the rates of outcome events that are count data, using Poisson regression or negative binomial regression.[5] We consider the same summary metric and atom metrics as above (all of these metrics are counts of events: game turns or uses of a card). As a concrete example, the system predicted that increasing card health from 1 to 7 would increase the rate of attacking with the card by 218%, aligning with the intuition that cards with more health can be used to attack more.

To build these models the system first checks for overdispersion, choosing a Poisson model when overdispersion was not detected and the negative binomial model otherwise.[6] The system then fits the corresponding model, producing incidence ratios that represent how a feature change alters the proportionate rate of the corresponding event.[7] Poisson and negative binomial regression are techniques that extend multivariate linear regression to predictions involving count data. The simpler Poisson model makes assumptions about the relative variability in the data, when there is more variability than assumed the data is overdispersed. In these cases the more general negative binomial model can be used (at the cost of greater model complexity). All ratios report increases compared to a default card configuration with 1 attack, health, and cost. For example, the value of $0.97$ for 'attack = 4' in Table II indicates that a card having an attack of 4 associates with games being $0.97$ the length of games where the card has an attack value of 1. Each coefficient also has a corresponding test of statistical significance: we only report features with significant ($p < 0.05$) effects, indicated in bold in the tables. Below we review a single key model for each metric considered.

*1) Summary Metric:* Of the three card parameters, attack and cost changed game length, while health did not (Table II). Cards cannot protect the player and thus card health is only useful for cards remaining on the board rather than extending game length. Intuitively, increasing attack power allows agents to more quickly defeat one another, while increasing cost slows down how quickly the card can be played and put to use, in turn lengthening the game.

*2) Atom Metric: Card Use:* In games like *Cardonomicon* the features of a card govern (human) choices to use different cards. Cards are often discussed in cost-benefit terms: weighing costs (e.g., mana cost) against benefits (e.g., card attack power or health) of playing the card. Modeling how changes to card parameters alter the choices to play or attack with a card helps designers understand how to alter a design to increase or decrease how often different card choices are made in the game. As such, we had the system model how changes to card parameters altered the frequency of attacking with cards or playing cards to the board (a pre-requisite for attacking).

[5]We used R's glm function for both: https://stat.ethz.ch/R-manual/R-devel/library/stats/html/00Index.html

[6]Checks used the dispersion test provided by R's AER package: https://cran.r-project.org/web/packages/AER/index.html

[7]We exponentiate the learned coefficients from the model to produce incidence ratios as these are more readily interpretable.

TABLE II
EFFECT OF CARD PARAMETERS ON GAME LENGTH

| Game length vs card parameters | |
|---|---|
| feature | coefficient |
| attack = 4 | **0.97** |
| attack = 7 | **0.94** |
| health = 4 | 1.00 |
| health = 7 | 1.00 |
| cost = 4 | **1.04** |
| cost = 7 | **1.04** |

TABLE III
EFFECT OF CARD PARAMETERS ON CARD ATTACK OR PLAY RATES

| Atom frequency vs card parameters | | |
|---|---|---|
| feature | play coef | attack coef |
| attack = 4 | **0.92** | **0.81** |
| attack = 7 | 0.94 | **0.74** |
| health = 4 | 1.04 | **1.55** |
| health = 7 | 1.00 | **2.18** |
| cost = 4 | **0.83** | **0.61** |
| cost = 7 | **0.48** | **0.27** |

Greater card costs significantly reduced frequency of play ('play coef' incidence ratio $< 1.0$), and subsequently reduce attack rates as well ('attack coef' incidence ratio $< 1.0$) (Table III). Thus, the system learned that more expensive cards are played less and consequently attack less. Greater health values increased attack rates: higher health allows cards to survive longer and consequently attack more. Greater card attack *reduced* attack rates: opponents with greater attack power eliminate one's own cards sooner, preventing attacks.

*3) Atom Metric: Card Options:* Card *use* is not the sole indicator of the influence of design parameters on play: variations in how often a card is an *option* for use can indicate how card parameters influence player behavior to use or hold on to cards. Card *options* are the cards an agent has the choice to use, either to attack an opponent or to play to the board. Unlike card actions (examined above), card options measure the strategic possibilities an agent has at hand. As before we had the system model how card parameters altered the frequency of having the option to attack with or play cards.

Contrary to card play *actions* (Table III), increased card cost reduced card play options (Table IV). Greater cost values increased the frequency of play options, reflecting that cost prevents a card from being played. Both greater card attack

TABLE IV
EFFECT OF CARD PARAMETERS ON CARD ATTACK AND PLAY OPTIONS

| Atom option frequency vs card parameters | | |
|---|---|---|
| feature | play coef | attack coef |
| attack = 4 | **0.94** | **0.71** |
| attack = 7 | **0.91** | **0.53** |
| health = 4 | 0.98 | **1.84** |
| health = 7 | **0.95** | **2.49** |
| cost = 4 | **1.45** | **0.71** |
| cost = 7 | **1.08** | **0.31** |

and greater card health led to small reductions in the rate of card play options. Thus, the system learned that increasing card benefits (attack and health) makes the card become more attractive to play, leading to fewer turns where the card is retained as an option. Card attack options showed similar trends to card attack actions. Greater card health increased attack rates, implying cards have strategic value: when a card has more health it is not only useful for the act of attacking, but also as an option for attacking later. Greater cost reduces the rate at which a card is available to attack as the greater cost gates use of the card.

## VI. Conclusion

We addressed automated design space modeling by using MCTS for behavior sampling, providing action metrics for play space gameplay analysis, and count regression models for design modeling. Below we discuss limitations of this work and future research avenues.

We studied a single set of techniques for behavior sampling, play space learning, and design modeling. Future work can investigate more sophisticated algorithms for each of these problems. Our behavior sampling algorithms are low-fidelity models of human-like play, intended to abstract and simplify human players as caricatures of typical players [24]. Behavior sampling may consider modeling human capabilities for memory, perception, or propensity for actions that are not utility maximizing (i.e., not 'rational' in the economic sense). These models offer the ability to capture broader ranges of player differences to better characterize how a game does (not) demand different human capabilities.

Our play space models intentionally focused on considering player actions alone, limiting our scope to a narrow set of design questions that are readily evaluated using player behavior: topics like game balance, pacing, or (a limited form of) strategic depth [39]. In most scenarios designers instead consider a wide range of player behaviors and responses to judge the success of a game. Play space models may combine action and state metrics to allow designers to compare both the decision and outcomes spaces of play. Other aspects of design have been ignored by the purely behavioral modeling choices here: for example player interpretation of game meaning or the messages conveyed by a game [40]; out-of-game communication between players (e.g., in games like *Diplomacy*); or subjective experiences like frustration, boredom, or curiosity. Developing techniques to connect these more subjective aspects of game experiences to game design features will be challenging, but is crucial to expanding these methods to a wider range of game design practices.

Our design models illustrate how predictive modeling can summarize a complex design space into simpler models that are more easily interpreted. We anticipate model improvements that improve the predictive powers of the models used or yield outputs that are more readily interpretable by humans. We only apply design models to optimizing low-level, continuously varying parameters of a design. Yet many aspects of a design are discontinuous: rule changes like card abilities, turn-taking structure, or victory/failure conditions. These types of spaces will require different models both for design optimization and modeling the space of design decisions. Future work can also explore ways to model higher-level design features (like game pacing or balance) and explore how to generalize and transfer design knowledge between game domains. Developing ways to aggregate and reuse design models has great potential to enhance the scope and capabilities of computational design systems.

Extensions of these algorithms may also explore other game design challenges and game domains. Our game domains are highly simplified in terms of player-player interactions: games like *Hearthstone* involve extensive reasoning on counters and chains. Scaling to this behavioral complexity will require efficient models to simulate behavior in wide decision spaces [41], potentially coupled with techniques like deep reinforcement learning to facilitate agents acquiring sufficiently advanced play skills [18]. Research in these areas will broaden the scope of design space modeling, extending this early work to more complex cases.

The models we developed have been applied to discrete, turn-based, perfect information, and adversarial games, leaving open many avenues for extensions and generalizations. This leaves open a wealth of alternative games and design problems to develop new techniques for. Extensions to new game domains include real-time games, continuous space games, hidden information games, or collaborative games. We anticipate scaling challenges to emerge across these domains, primarily stemming from the need to increase the amount of behavior samples generated when game action spaces increase in branching factor. Alternative game design domains include games based on social interaction or diplomacy among players (e.g., *Diplomacy* or *Cosmic Encounters*), games centered on conveying a message (*Dys4ia*) or inducing player contemplation and reflection (*Gone Home*), or games that mix digital and physical elements (*Pokemon Go*). Each of these new design domains will require extensions including more sophisticated simulations of player behavior and experience, and more general models of game design features. We believe the potential to accumulate and reuse design models across game design systems can accelerate the progress of these design tools, unlocking new opportunities for human and automated game design and development.

TABLE V
CARDS USED IN *Cardonomicon* EXPERIMENTS.

| Card Name | Health | Cost | Attack |
|---|---|---|---|
| Stonetusk Boar | 1 | 1 | 1 |
| Dire Wolf Alpha | 2 | 2 | 1 |
| Defias Ringleader | 1 | 2 | 2 |
| Kobold Geomancer | 2 | 2 | 2 |
| Ironfur Grizzly | 2 | 2 | 1 |
| SI Agent | 1 | 2 | 2 |
| Fen Creeper | 7 | 5 | 3 |
| Southsea Captain | 4 | 4 | 5 |
| Abusive Sergeant | 1 | 1 | 1 |
| Angry Chicken | 1 | 1 | 1 |
| Blood Imp | 1 | 1 | 1 |
| Super OP | 2 | 3 | 1 |
| Aldor Peacekeeper | 3 | 3 | 3 |
| Arcane Golem | 2 | 3 | 4 |
| Dalaran Mage | 4 | 3 | 1 |
| Dark Cultist | 4 | 3 | 2 |
| Scarlet Crusader | 1 | 3 | 3 |
| Ancient Brewmaster | 6 | 4 | 3 |
| Chillwind Yeti | 5 | 4 | 4 |
| Boulderfist Ogre | 7 | 6 | 6 |

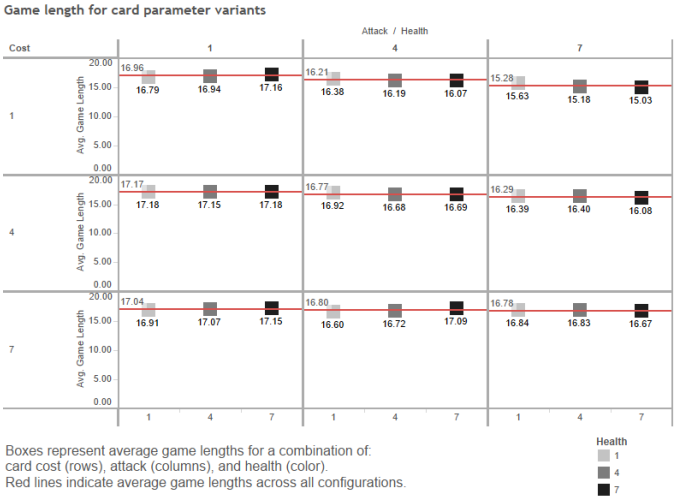


Fig. 8. Average game length based on card configuration.

## APPENDIX A
## *Cardonomicon* CARDS

Table V shows configurations of the cards used (1 each) in the decks of the *Cardonomicon* agents. Card settings were based on correspondingly named games from *Hearthstone*. The 1 attack, 1 health, 1 cost "Stonetusk Boar" card was the single card varied in the design learning experiment.

## APPENDIX B
## SUPPLEMENTARY DESIGN EVALUATION FIGURES

These figures visually support the learned design models above.

### A. Summary Metric

The effects of card parameters on game length are readily discernible when examining the average game length in each of the card parameter configurations. Figure 8 shows the average game length for different parameter configurations. Columns divide configurations hierarchically: first splitting by attack, then by health. Rows divide configurations by card costs. Red lines indicate values averaging over health values (the average length for a cost and health configuration). The decreasing average lengths for attacks (red lines) indicates that as attack increases, game length decreases. The higher average lengths for costs (rows) indicates that as cost increases, game length increases.

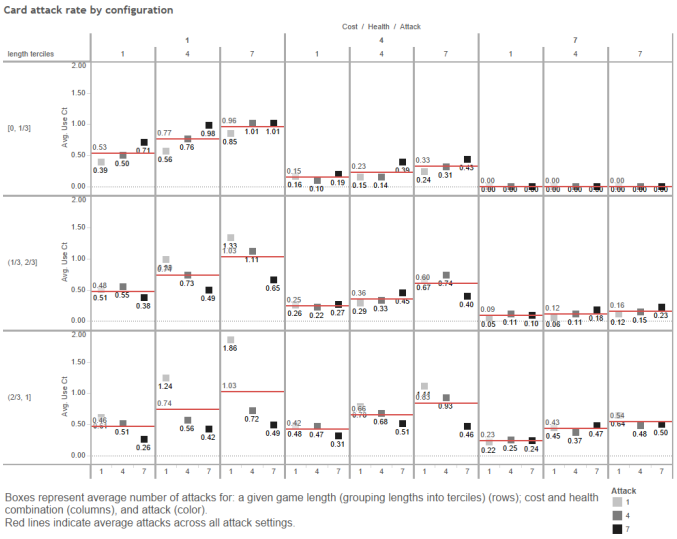


Fig. 9. Average number of times the card variant is used to attack given a card parameter configuration.

### B. Atom Metrics

*1) Card Attack Rates:* Figure 9 shows the average number of times the card variant was used to attack over games for different parameter configurations. Columns divide configurations hierarchically: first splitting by cost, second by health, and third by attack. Rows divide configurations by the game lengths, grouping games by length terciles (below 14 turns, between 14 and 16 turns, and more than 16 turns). Red lines indicate average card attack rate across attack configurations (marginalizing to game length, health, and cost combinations). High cost cards (far right triple of columns) reduce the frequency of card attack rates to near zero except in long games (bottom row). Increasing health (red lines) also increases card attack rates. These results visually corroborate the model learned by the system for human designer consumption.

*2) Card Play Rates:* Figure 10 is similar to Figure 9, only now displaying the average number of times the "Stonetusk Boar" card was played (rather than used to attack). Longer games allow more opportunities for play, seen by comparing the average play rates across the three game lengths (rows), especially in the highest cost scenario (far right column). Cost clearly reduces play frequency, seen by comparing the three sets of columns (cost is the outmost grouping of columns). Conversely, neither attack nor health appear to have a directional effect, seen by inconsistent relationships among play rates for different attacks (colors) or healths (red lines).

*3) Card Attack Options:* Figure 11 provides a visual overview of the card action option outcomes in a similar manner to Figure 9. The similarity to Figure 9 supports the

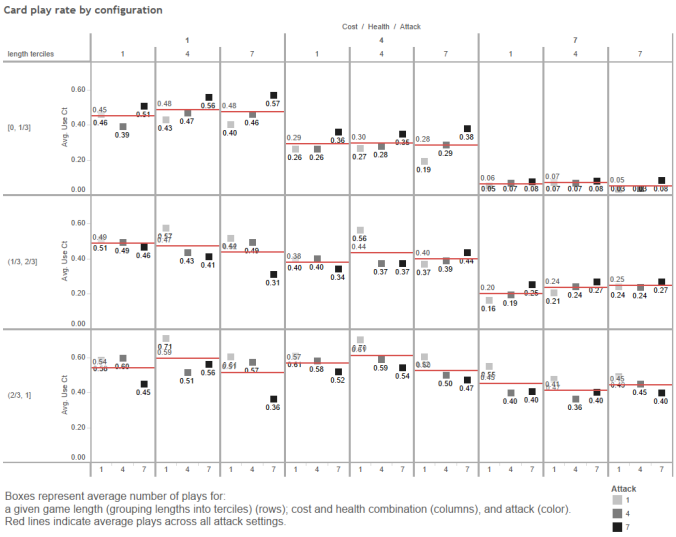


Fig. 10. Average number of times a card is played for a given card configuration.

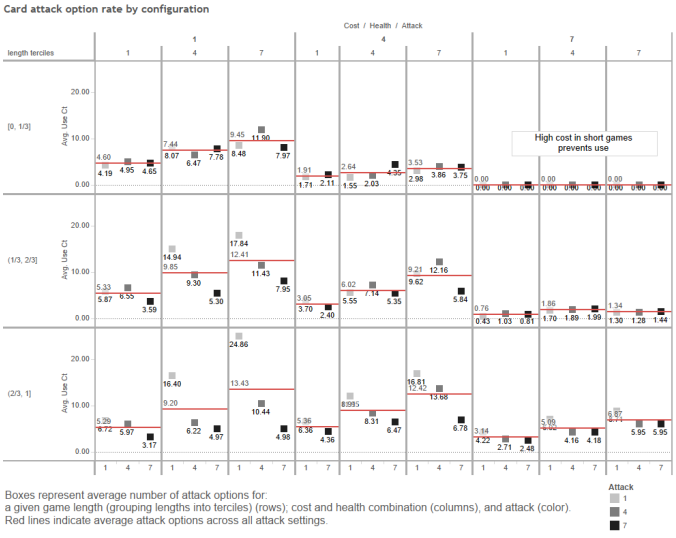


Fig. 11. Average number of options for card attacks by card configuration.

conclusion that card parameters have similar effects on attack actions and attack options.

*4) Card Play Options:* Figure 12 is similar to Figure 10, only now displaying the average number of times the card variant was an option to play, rather than being played. Longer games allow more play options, seen by comparing the average play rates across the three game lengths (rows), especially in the highest cost scenario (far right column). Cost clearly reduces play frequency, seen by comparing the three sets of columns (cost is the outmost grouping of columns). Conversely, neither attack nor health appear to have a directional effect, seen by inconsistent relationships among play rates for different attacks (colors) or healths (red lines).

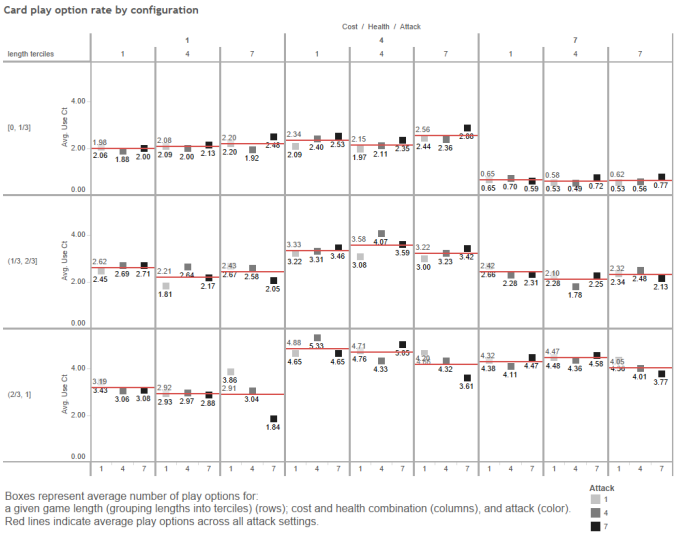


Fig. 12. Average number of options to play a card by card configuration.

## REFERENCES


[1] R. Hunicke, M. Leblanc, and R. Zubek, "MDA: A formal approach to game design and game research," in *Workshop on Challenges in Game AI*, 2004.

[2] T. Fullerton, C. Swain, and S. Hoffman, *Game Design Workshop*. Morgan Kaufmann, 2008.

[3] C. Macklin and J. Sharp, *Games, Design and Play: A detailed approach to iterative game design*. Addison-Wesley Professional, 2016.

[4] M. Seif El-Nasr, A. Drachen, and A. Canossa, Eds., *Game Analytics*. Springer London, 2013.

[5] A. M. Smith, M. J. Nelson, and M. Mateas, "Computational support for play testing game sketches," in *5th AAAI Conf. on Artificial Intelligence and Interactive Digital Entertainment*, 2009.

[6] ——, "LUDOCORE: A logical game engine for modeling videogames," in *IEEE Conference on Computational Intelligence and Games*, 2010.

[7] A. Zook and M. O. Riedl, "Automatic game design via mechanic generation," in *Proceedings of the 28th AAAI Conf. on Artificial Intelligence*, 2014.

[8] C. Martens, "Ceptre: A language for modeling generative interactive systems," in *11th AAAI Conf. on Artificial Intelligence and Interactive Digital Entertainment*, 2015.

[9] M. J. Nelson, "Game metrics without players: Strategies for understanding game artifacts," in *1st Workshop on Artificial Intelligence in the Game Design Process*, 2011.

[10] A. Isaksen, D. Gopstein, and A. Nealen, "Exploring game space using survival analysis," in *10th International Conference on the Foundations of Digital Games*, 2015.

[11] C. Browne, "Deductive search for logic puzzles," in *IEEE Conference on Computational Intelligence in Games*, 2013.

[12] E. Tomai, R. Salazar, and R. Flores, "Mimicking humanlike movement in open world games with path-relative recursive splines," in *9th AAAI Conf. on Artificial Intelligence and Interactive Digital Entertainment*, 2013.

[13] J. E. Laird and J. C. Duchi, "Creating human-like synthetic characters with multiple skill levels: A case study using the Soar Quakebot," in *AAAI 2000 Fall Symposium on Simulating Human Agents*, 2000.

[14] C. Holmgård, A. Liapis, J. Togelius, and G. N. Yannakakis, "Monte-Carlo Tree Search for persona based player modelling," in *First Workshop on Player Modeling*, 2015.

[15] S. Devlin, A. A., N. Sephton, C. P., and R. J., "Combining gameplay data with Monte Carlo Tree Search to emulate human play," in *12th AAAI Conf. on Artificial Intelligence and Interactive Digital Entertainment*, 2016.

[16] C. Browne and F. Maire, "Evolutionary game design," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 2, pp. 1–16, 2010.

[17] G. Elias, R. Garfield, and K. Gutschera, *Characteristics of Games*. MIT Press, 2012.

[18] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[19] G. Wallner and S. Kriglstein, "Visualization-based analysis of gameplay data – a review of literature," *Entertainment Computing*, vol. 4, no. 3, pp. 143 – 155, 2013.

[20] E. Andersen, S. Gulwani, and Z. Popović, "A trace-based framework for

analyzing and synthesizing educational progressions," in *ACM SIGCHI Conference on Human Factors in Computing Systems*, 2013.
[21] S. Björk and J. Holopainen, *Patterns in Game Design*. Cengage Learning, 2005.
[22] A. Summerville, S. Snodgrass, M. Guzdial, C. Holmgård, A. K. Hoover, A. Isaksen, A. Nealen, and J. Togelius, "Procedural content generation via machine learning (PCGML)," *ArXiv e-prints*, 2017.
[23] G. Smith and J. Whitehead, "Analyzing the expressive range of a level generator," in *1st Workshop on Procedural Content Generation in Games*. ACM, 2010, p. 4.
[24] A. Smith, C. Lewis, K. Hullett, G. Smith, and A. Sullivan, "An inclusive view of player modeling," in *6th International Conference on the Foundations of Digital Games*, 2011.
[25] N. Shaker, J. Togelius, and G. N. Yannakakis, "The experience-driven perspective," in *Procedural Content Generation in Games*. Springer, 2015.
[26] A. Summerville, M. Guzdial, M. Mateas, and M. Riedl, "Learning player tailored content from observation: Platformer level generation from video traces using LSTMs," in *Experimental AI in Games Workshop 3*, 2016.
[27] M. Guzdial, N. Sturtevant, and B. Li, "Deep static and dynamic level analysis: A study on Infinite Mario," in *Experimental AI in Games Workshop 3*, 2016.
[28] J. Togelius and N. Shaker, "The search-based approach," in *Procedural Content Generation in Games*. Springer, 2015.
[29] J. Togelius, G. Yannakakis, K. Stanley, and C. Browne, "Search-based procedural content generation: A taxonomy and survey," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 3, no. 3, pp. 172–186, 2011.
[30] N. Sturtevant, "An argument for large-scale breadth-first search for game design and content generation via a case study of fling!" in *2nd Workshop on Artificial Intelligence in the Game Design Process*, 2013.
[31] G. Chaslot, J.-T. Saito, J. W. H. M. Uiterwijk, B. Bouzy, , and H. J. van den Herik, "Monte-Carlo strategies for computer Go," in *18th Belgian-Dutch Conference on Artificial Intelligence*, 2006.
[32] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, "A survey of monte carlo tree search methods," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 4, pp. 1–43, 2012.
[33] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine learning*, vol. 47, no. 2-3, pp. 235–256, 2002.
[34] A. Jaffe, "Understanding game balance with quantitative methods," Ph.D. dissertation, University of Washington, 2013.
[35] A. Zook, B. Harrison, and M. O. Riedl, "Monte-Carlo tree search for simulation-based strategy analysis," in *10th International Conference on the Foundations of Digital Games*, 2015.
[36] D. Cook. (2007, 7) The chemistry of game design. website. UBM Tech. [Online]. Available: http://www.gamasutra.com/view/feature/1524/the_chemistry_of_game_design.php
[37] S. Tekofsky, P. Spronck, A. Plaat, J. van den Herik, and J. Broersen, "Play style: Showing your age," in *IEEE Conference on Computational Intelligence and Games*, 2013.
[38] T. Cadwell, "Counterplay and teamplay in multiplayer game design," in *Game Developers Conference*, 2013. [Online]. Available: http://gdcvault.com/play/1018158/Counterplay-and-Teamplay-in-Multiplayer
[39] F. Lantz, A. Isaksen, A. Jaffe, A. Nealen, and J. Togelius, "Depth in strategic games," in *AAAI Workshop on What's Next for AI in Games?*, 2017.
[40] C. Martens, A. Summerville, M. Mateas, J. Osborn, S. Harmon, N. Wardrip-Fruin, and A. Jhala, "Proceduralist readings, procedurally," in *Experimental AI in Games Workshop 3*, 2016.
[41] S. Ontañón, "The combinatorial multi-armed bandit problem and its application to real-time strategy games," in *9th AAAI Conf. on Artificial Intelligence and Interactive Digital Entertainment*, 2013.

**Alexander Zook** Alex Zook is a Senior Data Scientist at Blizzard Entertainment with a PhD from the Georgia Institute of Technology. His research focuses on ways intelligent systems augment the game design process. He applied artificial intelligence and machine learning to modeling game players, supporting human game designers, and automating routine parts of game design both in academic research and industry practice.

**Mark O. Riedl** Mark Riedl is an Associate Professor in the Georgia Institute for Technology School of Interactive Computing. His research spans the areas of artificial intelligence and machine learning with a focus on computer games, storytelling, and virtual worlds.