Game Balancing in Dominion: An Approach to Identifying Problematic Game Elements

Cassandra Ford,[1] **Merrick Ohata**[2]
[1] Lafayette College
[2] Johns Hopkins University
fordcb@lafayette.edu, mohata1@jhu.edu

Abstract

In the popular card game Dominion, the configuration of game elements greatly affects the experience for players. If one were redesigning Dominion, therefore, it may be useful to identify game elements that reduce the number of viable strategies in any given game configuration - i.e. elements that are unbalanced. In this paper, we propose an approach that assigns credit to the outcome of an episode to individual elements. Our approach uses statistical analysis to learn the interactions and dependencies between game elements. This learned knowledge is used to recommend elements to game designers for further consideration. Designers may then choose to modify the recommended elements with the goal of increasing the number of viable strategies.

Introduction

Dominion is a popular deck-building game that has a set of cards (distinct from a standard 52-card deck) as a collection of game elements. Before a single game begins, the players choose some number of the elements to include in the game, and which to exclude. They collectively choose 10 elements to be in the “Kingdom”, in addition to a static set of 7 elements which are always included (Copper, Silver, Gold, Estate, Duchy, Province, Curse). Each player begins with their own copy of the same deck (7 Copper and 3 Estates), and on each turn they can “buy” elements from the kingdom. Those elements are then incorporated into their deck and can then be played on subsequent turns with the goal of acquiring victory points (Dom 2016). The Kingdom, then, has a great impact on the number of viable strategies accessible to players during gameplay, since gained elements affect what actions may be taken by the player. We refer to a game configuration to be a kingdom, and a game as a single simulation run with that configuration.

When players buy or use elements, it affects the reward a player will receive in their effort to win the game (in this case, the number of victory points gathered). We can think of elements as possessing individual reward distributions, which are affected by the presence of other elements within a configuration, since it is true for many games that the presence of certain elements impacts how well other elements perform. This is similar, for example, to random resources generated at the beginning of a Fortnite match or ranking of a sprinter between heats - while an element may individually perform a certain way in one setting, in the presence of other elements it may perform differently. We therefore aim to model the overall distribution of each element individually across all situations, while also gaining insight into the interactions between elements that affect the reward output.

When formulating a model, Dominion can be thought of as a variation of an n-armed bandit problem. In this classic problem, when a lever on the bandit is pulled, some reward chosen from an unknown probabilistic distribution is given. A strategy is defined as some function determining which levers are pulled and the order to pull them in, and finding an optimal strategy for the bandit is a common problem in machine learning.

A typical method of solving this problem is learning the reward distributions of each lever. In Dominion, each game element can be thought of as one of the arms of this bandit - and it is assumed that the goal of the game designers is to create a set of game elements which enable numerous viable strategies. A viable strategy is one that some intelligent player could reasonably use to achieve the most reward over its opponents. We reason that the maximum number of viable strategies exists when each element is played an equal number of times by an intelligent player over the course of many individual games. It is not necessary that a viable strategy include many elements, but rather that any element exists in some viable strategy. Thus the goal of our research is to identify game elements that reduce the number of viable strategies.

By learning the distribution of rewards of the game elements, we can identify elements that reduce the number of viable strategies. In order to learn the distributions, we also need to be able to assign credit (and thereby rewards) to elements based on their influence on the outcome of the game. We can then flag certain elements as potential limiting factors in the number of viable strategies to the game designer.

We assume the goal of the game designers is to create what is referred to as a ‘balanced’ game. For our research, we interpret balance to mean having elements with similar reward distributions, so that no element is dominant over any other element. A better understanding of which elements negatively impact this aspect of play can help designers adjust element mechanics to achieve a more balanced game. Unfortunately, even with extensive playtesting, it can be very difficult to predict and identify which elements will create an unbalanced game environment before release to the public. Better balance is often achieved by ”nerfing”, ”buffing”, or ”banning” certain game elements post release (Izaak 2020; MTG 2021). We therefore aim to create a system which will indicate to designers which elements may need to be adjusted or removed.

Background

Other explorations on evaluating and balancing Dominion have already been made. One by Ransom Winder utilized several neural networks to evaluate the game state. His research took into account different mechanics-based parameters to help the networks improve strategic play. He also provided a general metric for evaluating cards by considering the average number of copies in decks of the learned players over the course of games against a player with a baseline strategy (Winder 2014). Another paper by Mahlmann, Togelius, and Yannakakis, included the testing of several AI players against each other. In their model, they seek to identify configurations that maximize different metrics of balance and enjoyability. They found that certain cards inherently balance the game regardless of strategy (Mahlmann, Togelius, and Yannakakis 2012). We look to expand on this research by examining the balancing influence of individual elements and, critically, interactions between pairs of elements in an effort to help game designers understand the underlying structure of their game.

In preparation for tackling this problem, Bayesian models (Brownlee 2019) particularly interested us as an effective way to represent the interactions between elements in the system. We could model each game element as having some reward distribution associated with it, modeling its positive effect on a player’s outcome of the game. It is true of many different games that elements have interactions with one another, that the presence or use of an element A impacts the effectiveness (or reward distribution) of another element B. We thought of modeling this as a Bayesian Belief Network.

A Bayesian Belief Network operates by mapping a set of conditional probabilities onto a graph, where nodes are random variables and edges describe dependencies. We thought modeling the set of game elements as a Bayesian Belief Network could result in more detailed predictions for the reward distributions and thus also predict how balanced a given configuration will be, even if that configuration had not yet been tested. This could help in discovering elements that are out of balance, or which cause certain game configurations to be out of balance. For instance, if the predicted influence of an element is much lower than the other elements in a large portion of possible game configurations, then it may be reasonable to recommend this element for further consideration.

We realized, however, that this model would ultimately lead to an issue of parsing, since elements could easily have a feedback effect and result in cyclic dependency. Cyclic Bayesian Belief Networks have yet to be solved, as there is no consistent way to parse such a network(Tulupyev and Nikolenko 2005). Some workarounds exist, and in our case, we would linearize the network by allowing each child to only be parsed once. Because of these problems, the Bayesian Belief network has not yet been incorporated into much of the approach, although we speculate on its potential uses throughout the paper.

We also discovered a Master’s thesis by Jon Vegard Jansen and Robin Tollisen, which we believe to be the most advanced dominion AI to date (Jansen and Tollisen 2014). The Agent uses Monte Carlo Tree Search (MCTS) to make decisions. Monte Carlo methods make decisions based on a number of random playout simulations, and Monte Carlo tree search does so while expanding the search tree to make more informed decisions. We used this Agent to perform our experiments, although the algorithm of choice is important only in that it produces intelligent play.

Problems

It is necessary to be able to identify elements that greatly reduce the number of viable strategies for any given game configuration. We consider elements that do this to be problematic. Variance between elements’ reward distributions is expected given the probabilistic nature of the game. However, it is undesirable to have distributions with centers well above or well below the other reward distributions as that would make certain elements unusable or dominant, both of which would skew element usage away from the desired state. We must be able to distinguish between these effects. However, we must first be able to identify an element’s reward for how much influence over the outcome it had for any given episode before we can identify any as problematic.

It is also important to consider the fact that game elements do not exist independently. We assume that by using one element, it may affect other elements’ reward distributions. This could lead to groups of elements which, while individually unproblematic, create feedback loops that dramatically increase or decrease the reward obtained. These relationships can take many forms, but by understanding the underlying interaction mapping of the elements, game designers may be able to better decide how to make adjustments mechanically to balance the reward distributions. The discussion of game elements as too powerful outside of the context of their relationships to other game elements is meaningless, as these elements are made powerful or weak by the context of their use. We therefore must consider how to identify these interactions in the solution. For this research, we only address interactions between pairs of elements. However, our approach described below can be used to test groups of three or more elements, as it is conceivable that many elements could simultaneously interact with one another.

Since our goal is to identify and recommend elements to developers that may reduce the number of viable strategies, we must also have a metric to determine when we know enough about the elements to claim problematic status. Without such a mechanism, the system may falsely flag elements as problematic due to variation in performance between configurations.

Approach

All data was collected using an artificial player utilizing a Monte Carlo Tree Search Algorithm (Jansen and Tollisen 2014) against a control opponent. The control used a strategy referred to as "Big Money", which only utilizes the static set of elements that are always included (and which are beyond the scope of our evaluation). The game ends when one of several game-ending conditions has been reached, and the player with more "points" in their deck is determined the winner (Dom 2016).

For this domain, we determined that the reward distributions for elements could appropriately be interpreted as influence evaluations. For our purposes, we assume that the distribution of average evaluations of all game elements is approximately normally distributed, though the individual reward distributions for each element individually need not be. Because of our definition of an episode and the size of our data set, we find it reasonable to assume that the mean sampling distribution of evaluations for each element over an episode is approximately normal. Thus most of our analysis consists of manipulating these sample means.

Experimental Design

A game configuration is defined by some subset of all game elements. An episode is defined to be 30 individual game simulations using the same game configuration. For data collection, we ran many episodes using this virtual player.

In the real domain, a game configuration must contain exactly 10 elements. To better limit the scope of the Agent and to avoid problematic elements skewing other distributions by their over-inclusion, we decided to modify this rule such that a game configuration could contain any number of elements. This allows the Agent to exclude certain elements that may prevent furthering our understanding of other elements. It also allows the Agent to gain a cursory perspective on the general performance of elements when many or all are present.

Our first 200 episodes were run using a brute force approach for designing the game configurations. i.e., running all possible configurations for some number of included elements. For the last few episodes, we ran them using a Planner that attempted to gain more information on what we believed to be poorly understood elements. The effect of this Planner is addressed further in the discussion.

Evaluation

We have a heuristic function to evaluate the influence of any element over the outcome of the episode. This function is key to the entire analysis, so we assume that it is in fact a good evaluation of an element's influence over the outcome of an episode. We can then use this heuristic evaluation to hypothesize on future configurations. This process is sketched in Algorithm 1 below.

For our domain, we determined that the best measure of influence would be based on the number of times an element was used, as this corresponds directly to its effect on the game. Since elements have no inherent influence over the game if they aren't used, this was the most common sense approach. For each player, we thus included a term for the number of times the element was used. We decided to divide this value by the total number of turns taken to prevent the artificial inflation of evaluations for elements that encourage longer games. We also divide by the number of copies of the element in the deck to take into account that additional uses are possible with more copies. Additionally, since the goal of the game is to gain victory points, we found it reasonable to take the average of the uses-over-turns-per-copy value for each player weighted by the number of victory points possessed at the conclusion of the game. The purpose of this was to obtain a single value that would reflect the effectiveness of an element in helping the player achieve the goal of winning the game. For games where an element was used many times by a player that did not gain a large portion of the awarded points, the large number of uses will be appropriately scaled by the weighting; on the other hand, if an element was used many times by a player who succeeded in gaining the majority of points, the element's role will also be scaled by the weighting.

We also needed to address the issue of elements which could be included in a configuration and added to a player's deck, but which cannot be used due to the nature of the element (in particular, we needed a way to evaluate the element Gardens). To account for this, we also include a weighting based on how many copies of an element was added to the winning player's deck. In this way, no credit is assigned to unplayable elements in the losing player's deck, but is acknowledged if contributing to victory.

In equation (1) below for E, the evaluation of a single element for a game, s_i, n_i, and p_i are the number of victory points, copies of the element in the deck, and uses of an element respectively for player i over the course of a single game. T is the total number of turns the game took. Note that in the last expression, $v = i$ of the winning player.

$$E = \frac{s_2}{s_1 + s_2}\left(\frac{p_1}{T \cdot n_1}\right) + \frac{s_2}{s_1 + s_2}\left(\frac{p_2}{T \cdot n_2}\right) + \frac{s_v}{s_1 + s_2}(n_v) \tag{1}$$

It is important to mention that for our experimental design, one of the players never plays elements nor includes them in its deck, thus both n_i and p_i are zero for one of the players i. In this case of a $\frac{0}{0}$, the undefined expression is instead set to zero.

Hypothesis Generation

We accumulate data from many previous simulations and build upon our knowledge of the elements. The accumulated data is used to describe each element's sample distribution to estimate the true reward distribution of that element. We also examine the relationships between these distributions to determine if an interaction exists. These sample distributions and the interactions define our hypothesis.

We assume that our hypotheses are accurate once the difference between the predicted distribution and the observed reward reaches a lower threshold. If it is consistently reasonable that the observed reward came from the predicted distribution, we can be more confident in claiming that the flagged elements are indeed problematic. In the future, the Bayesian

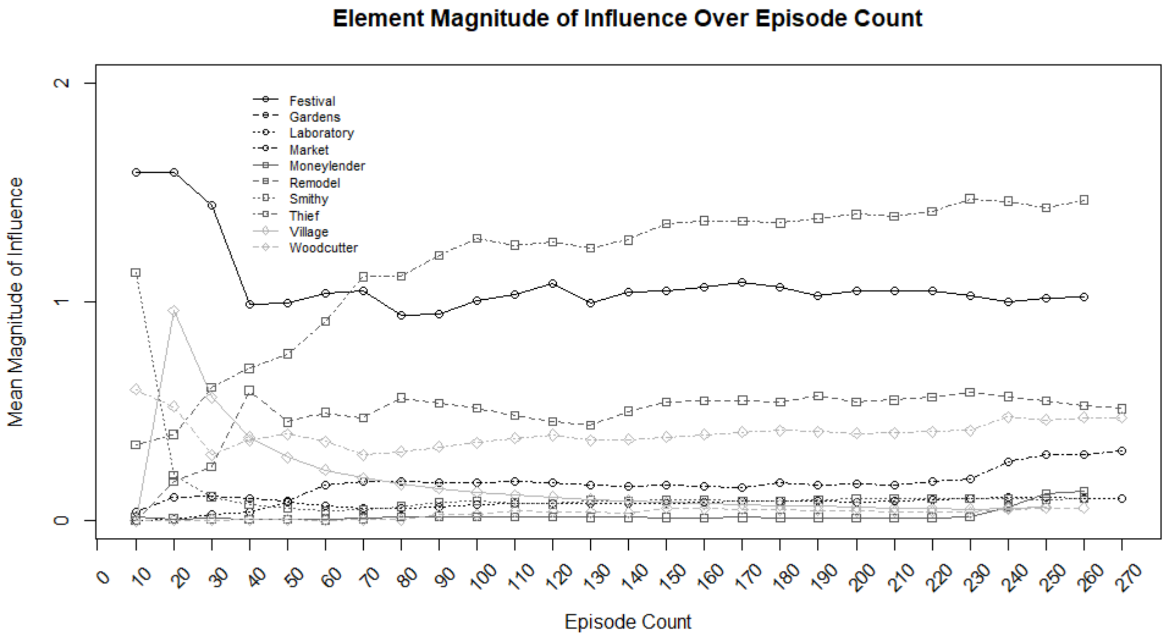

Figure 1: μ of reward distributions without problematic elements

```
Algorithm 1: Hypothesize
Input: Episode metadata, predicted reward distribution
    •Add data to database
    •Using all accumulated data, find the distribution of each element
    •Do a $t$-test of each element against the set containing every other element
    •Do a $t$-test of each element pairing against the set containing every other element
    •Construct a new Hypothesis set as (Reward Distribution, Interaction Mapping, Problematic Elements)
    •Plan the new set of game configurations based on current Hypothesis set
Output: New Hypothesis Set, Set of Game Configurations
```

Belief Network (described more below) would provide valuable aid in forming the hypotheses, as it takes into account the effect of elements on one another.

In order to identify whether an element is problematic, we examine our current hypothesis. We assess the probability that an element's mean evaluation comes from the distribution of average evaluations from all other elements. If the probability is below some threshold determined by the game designer, it is flagged for further analysis. We can also identify whether pairs of elements are problematic by running the same procedure on the average of the two elements and treating it as a single entity. Since the average is a linear combination of two approximately normal random variables, this is a valid test. A similar approach can be taken when considering trios or larger groups of elements as well, though these groupings were not addressed in this research.

Algorithm 1 further explains the process described above used to generate the new hypothesis, which is then used by the Planner (described in the next section) to create a new set of experiments to run.

Planning

In the real domain of Dominion, there are $\binom{24}{10}$ different game configurations for the base set alone and it is unreasonable to attempt to play all of them. Additionally, there have been numerous expansions released since the first edition of the game, and it is unreasonable to attempt to play all of the combinations available between them. We therefore created a program determining new sets of game configurations to test, which we called the Planner.

It is desirable to have a data set that reaches all corners of the topography of the estimated function, so that given some game configuration, we can be reasonably sure the algorithm has a good understanding of that area. This is achieved by having experiments in the same region as any given game configuration. If two game configurations are very similar, maybe having only a single element different between them, we can be reasonably sure that the data from one will be relevant to the other.

Given the large number of combinations associated with our configuration set, it is impossible to fully explore every possible combination. Therefore, based on the available data, it is desirable to decide where poorly understood regions of the estimation function are, and to run experiments on game configurations in those regions.

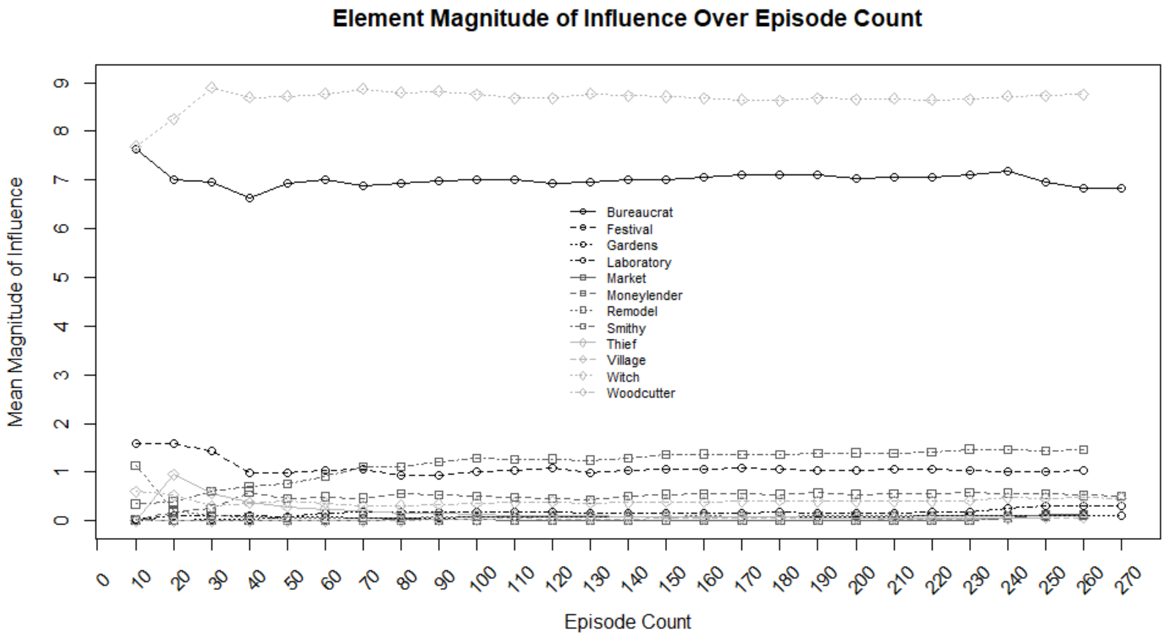

Figure 2: μ of reward distributions with all elements

The Planner selected configurations to include in subsequent testing by predicting the outcome of potential configurations and evaluating the soundness of the prediction. If any of the distributions included in the configuration were particularly ill-defined (due to high variance or little data, for example), then it was determined to require further testing.

Interactions

For identifying pairs of interacting elements, we perform a two-sample t-test using the set of evaluations from game configurations when only one of the two elements was present and the set of evaluations when both elements were present. This test is performed for each element in the pair. Our null hypothesis is that the means of the two sets of data will be equal. If we find the t-statistic to fall above or below a certain threshold, we can say that there is an interaction between the elements. The results of this testing also implicitly defines a Directed Graph (element A interacts with element $B \to A \to B$) and lays the foundation for a Bayesian Belief Network model of the interactions between elements. Although this testing only addresses pairs of elements, we can extend the same procedure to test interactions between larger groups by considering each two-component partition in the set.

In the future, the Bayesian Belief Network could be used to more accurately predict the specific reward distribution for an element in any given game configuration. The distributions of the interconnected game elements define a k-dimensional reward distribution, where k is equal to the number of game elements being considered. To more accurately identify problematic elements, we consider this k-dimensional space. The network could then help identify undesired peaks and valleys of this function by giving more accurate assessments of specific reward distributions of elements in conjunction with one another. This would render the flagging of potentially problematic elements more discriminant. It would also aid the discovery of unexplored regions for the Planner, and also better identify when we do not understand a region of the function well. This is because it has more sophisticated parsing than our current naive approach. The Planner can then use the information gathered by the network to generate new game configurations.

Results

Element Distributions

Based on the program data, the elements Witch and Bureaucrat were both consistently flagged as problematic. The dramatic difference in evaluations of these two elements can be easily observed in Figure 1. While all other elements fell within a similar range of evaluation (or magnitude of influence) between near-zero and around 1.6 (see Figure 2), both Bureaucrat and Witch consistently had scores that were four or more times higher than the maximum in this range once the scores began to converge.

Interaction Mapping

Our interaction mappings are represented as graphs. Figure 3 shows the interactions which were identified after 100 episodes, and Figure 4 illustrates the more refined set of interactions identified by the conclusion of testing. In the graphs, each node is a game element, and a directed edge

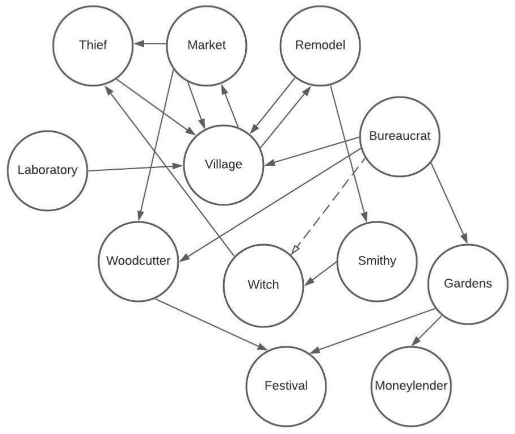

Figure 3: Mapping after 100 episodes

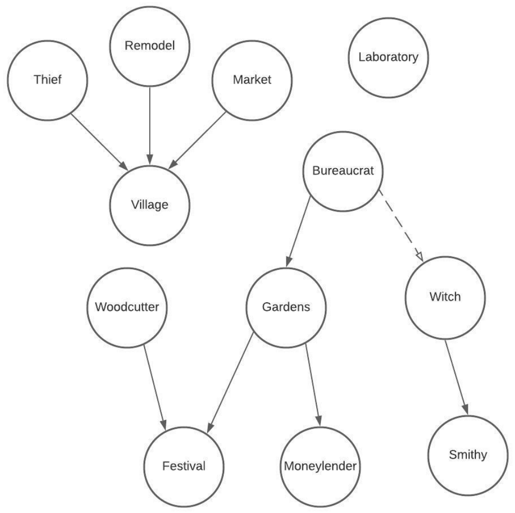

Figure 4: Final Mapping

represents a relationship where the source affects the related node, which is either a positive or negative relationship. We call a relationship problematic if either of the adjusted reward distributions is considered problematic using the same test as individual elements. In the interaction mappings, problematic relationships are represented with a dotted line and hollow arrow.

According to our data, the elements Bureaucrat and Gardens have a positive relationship, with Bureaucrat raising the average reward distribution of Gardens by over 7 standard deviations. This is an example of mechanical synergy between elements.

Our most problematic elements, Bureaucrat and Witch, have a negative relationship where Bureaucrat lowers the reward for Witch. This is an instance where a problematic interaction has more to do with the elements within the interaction than the interaction itself.

Discussion

The primary discovery of our data is that the elements Bureaucrat and Witch are much more powerful than the other elements. We find our results to be reasonable for our collection of game elements and based on our knowledge of the domain and other player-sourced element evaluations (markus 2021) (Gli 2020). If we exclude these problematic elements, we also observe the elements Festival and Smithy are well above the others (see Figure 2). While these two elements were not considered problematic, given the interconnected nature of the domain, a game designer may wish to further investigate these elements when considering adjustments to other problematic elements.

Our initial mapping is a lot more chaotic than our final mapping, and we believe the mapping would continue to narrow as more data was accumulated. It is our belief that the interaction mapping, and an accurate Bayesian Belief Network, will take far more data to converge than the ranking of elements. The method itself does seem to be sound and is identifying interactions that seem reasonable based on the mechanics of the elements. In the aforementioned example of Gardens and Bureaucrat, we find this interaction to be reasonable given that Gardens gives a player points based on the number of cards in that player's deck and Bureaucrat adds cards to the deck when used.

The network mapping could be helpful in use with the Planner. By having a more accurate understanding of the system as a whole, it can be used to find gaps in knowledge more accurately than without one. While we weren't able to generate results of the Bayesian Belief Network in use, it should be noted that cycles can (and frequently do) exist in this mapping. As discussed in the introduction, resolving cycles in Bayesian Belief Networks is an open problem, and while workarounds exist, none are true solutions.

==

Potential Bias

There is a great deal of bias introduced into this data. We first address the fact that all data was collected examining a set of twelve elements, which may seem a relatively small sample. The primary reasoning behind this decision was simply due to time constraints. The relatively rapid timeline for this project from conception to completion made the implementation and testing of additional elements unfeasible. According to our understanding of the problem, however, the number of elements included should not be an issue, since each element and pair of elements are only considered against each other. Including more elements would make collecting enough accurate data harder, but the data analysis would remain fundamentally the same.

The Planner was introduced near episode 220, and the movements in the distributions past those episodes speaks to the introduced bias of the Planner (see Figures 1 and 2). After episode 140, the evaluations of elements appear to stabilize around constant values. However, after 220, the evaluations begin to move steadily away from this steady state. This indicates that the way game configurations are selected by the Planner affect the observed reward distributions, and is thus a source of bias. In future implementations of this approach, any Planner should be carefully designed.

Our approach also assumes that one has access to large amounts of intelligent play data. Unintelligent play would not necessarily produce incorrect evaluations depending on the heuristic evaluation function. However, in our domain, an element with a high reward distribution is only as rewarding as the player identifies it to be. In this domain, an element can only be exploited; if a player does not choose to use it, it has no inherent influence over an episode. This creates a large bias problem. If there exists numerous viable strategies but our player favors any single one in particular, we may falsely identify elements as problematic, when in fact it was the player causing this problem, not the elements.

We must finally consider the Player as a source of bias. As previously asserted, game elements have no inherent influence on a game, and so if an element is not used it has no influence, and receives no reward. If the player, for some reason, decides not to use a powerful element, or instead prioritizes a weak element, this could give a reward that may be disproportionate to the true reward distribution. The player we used used a Monte-Carlo search algorithm, and thus included a random element. This made multiple simulations of the same configuration meaningful, and gave the variance that partially addresses this problem. All data must be considered in the context that the player is largely what makes it accurate or inaccurate.

Future Work

Our work on this problem has led us to consider further work in investigating recommendation systems for the game designers. For this particular domain, the most obvious further research would be to introduce the remainder of the base set into the system with the possibility and intention of including expansion sets as well. With this dramatically increased element pool, the system would have greatly reduced bias in its evaluations as it would take more elements into account. We could also gain a better understanding of the complex interactions between elements through our mapping, though, as discussed above, this would require vast amounts of testing. For this system, or an expanded iteration of it, we would also like to investigate different planning approaches. As we mentioned before, the Planner introduced a great deal of bias into the system, thus looking for ways to minimize this effect is a natural continuation.

Another further project would be the formulation and testing of multiple heuristic evaluation functions. Due to time constraints, we simply developed a function which we determined suited our purposes and used it for all of our evaluations. We could also, however, test multiple related functions in an effort to identify problematic elements in fewer iterations by looking at many different metrics from play data. In a similar vein, we could also explore measuring reward according to an ideal other than balance in influence; for example, we may wish to find cards and pairs which lead to extremely long turns or very short games. A related but equally intriguing possibility is the evaluation of "balance" itself or other desirable qualities over the course of a game, which could lead to recommendations of enjoyable configurations for players.

Conclusion

While designing a game, it is desirable to balance it such that no game element restricts the number of viable strategies available to an intelligent player. We developed a tool to help identify elements that have this effect to aid in the development process. The main improvements our system could benefit from is a more well-rounded player, a more comprehensive heuristic evaluation function, and a more intelligent Planner. The ability to assign credit to elements within a complex interacting system and to identify interactions between those elements is a well-studied problem. Our approach may be applied to other domains with complex interacting systems where the effect of elements is largely unknown.

References

2016. Dominion 2nd Edition Rules. https://www.riograndegames.com/wp-content/uploads/2016/09/Dominion-2nd-Edition-Rules.pdf. Accessed: 2021-12-28.

2020. Dominion Card Glicko. http://wiki.dominionstrategy.com/index.php/Dominion_Card_Glicko. Accessed : 2021-12-28.

2021. Banned and restricted cards Timeline. https://mtg.fandom.com/wiki/Banned_and_restricted_cards/Timeline. Accessed: 2021-09-09.

Brownlee, J. 2019. A Gentle Introduction to Bayesian Belief Networks. https://machinelearningmastery.com/introduction-to-bayesian-belief-networks/. Accessed : 2021-06-20.

Izaak. 2020. Fortnite: Patch 13.20 weapon buffs and nerfs. https://www.sportskeeda.com/esports/news-fortnite-weapon-buffs-nerfs. Accessed: 2021-09-09.

Jansen, J. V.; and Tollisen, R. 2014. *An AI for dominion based on Monte-Carlo methods*. Ph.D. thesis, University of Agder.

Mahlmann, T.; Togelius, J.; and Yannakakis, G. N. 2012. Evolving card sets towards balancing dominion. In *2012 IEEE Congress on Evolutionary Computation*. IEEE.

markus. 2021. Dominion Card Glicko. https://docs.google.com/spreadsheets/d/1CaVOd1pgAgmjJHXPM1tVMVnlJOLDZaq8BxjW4I1NI1E. Accessed: 2021-12-28.

Tulupyev, A.; and Nikolenko, S. 2005. Directed Cycles in Bayesian Belief Networks: Probabilistic Semantics and Consistency Checking Complexity. In *MICAI 2005: Advances in Artificial Intelligence*, 214–223. ISBN 978-3-540-29896-0.

Winder, R. K. 2014. Methods for approximating value functions for the Dominion card game. *Evol. Intell.*, 6(4): 195–204.