Identifying Strategies in *Dominion* Using Playtrace Clustering

Anthony Owen

***Abstract*—We demonstrate the use of playtraces and playtrace clustering to identify strategies and card synergies in deck building card games, using *Dominion* as an example. We analyze playtraces generated from both online human play and a variety of AI agents, examining two types: card counts by round in a player's deck and N-Grams generated from player actions. Using both the L_k-norm and Jensen–Shannon distance measures, in-conjunction with K-Means, K-Medoids and DBSCAN algorithms, we show that playtraces and distinct clusters can reveal both longer term strategies and card synergies. In addition, we use a restricted play framework to increase the variation in strategies and tactics explored by AI agents. Finally, we suggest that the game-agnostic, N-Gram-based approach may support strategy exploration in tabletop games more broadly.**

***Index Terms*—Board games, clustering, deck building card games (DBCGs), Monte Carlo tree search (MCTS), playtraces.**

I. Introduction

DECK Building card games (DBCGs) are a genre of modern card games that began with *Dominion* (Rio Grande Games, 2008). These games feature unique set of cards and include titles, such as *Ascension* (Stone Blade Entertainment, 2010), *Star Realms* (Wise Wizard Games, 2014), *Star Wars: A deck building game* (Fantasy Flight Games, 2023), and many others.

In DBCGs, players manage cards in their hand, draw pile, and discard pile. On a player's turn, they play cards to generate resources or effects and purchase new cards from a central supply, adding them to their discard pile. When a draw pile is exhausted, the discard pile is shuffled into a new draw pile, evolving the player's deck. Effectively the player is designing an engine through card choices to achieve victory.

Elements of deck building card games have also been integrated into board games and video games. In the board game *Clank!* (Renegade Game Studios, 2016), cards are used for board movement and combat. In the video game *Slay the Spire* (Mega Crit, 2017), battles involve drawing and playing cards, with players selecting one of three cards after each battle to evolve their deck over time. In addition, collectible card games (CCGs) are close cousins to DBCGs, with deck-building occurring prior to gameplay. Classic examples of CCGs include *Magic: The Gathering* (Wizards of the West Coast, 1993), *Pokémon Trading Card Game* (Media Factory, 1996), and *Hearthstone* (Blizzard Entertainment, 2014).

A key challenge of DBCGs (and CCGs) is that the impact of each card can depend on which other cards exist in a player's deck or hand. This creates a combinatorial problem, For instance, a game with 100 distinct cards and a typical deck size of 20 yields approximately 5×10^{20} possible decks. This makes identifying strategies, synergies, and balancing the game both time-consuming and difficult.

An approach to this challenge is to use the concept of a playtrace. A playtrace is a data driven record of the actions, behavior or state of a player during the course of a game. Repeated play of a game generates a set of playtraces, and a distance measure that quantifies the similarity between a pair of playtraces can be used to group or cluster similar playtraces together. These clusters can then be analysed for insights into gameplay.

This article demonstrates that using playtraces and playtrace clustering from human and AI simulated play of deck building card games can provide a means to identify available strategies and tactics. Using *Dominion* as a case study, we analyse two different types of playtraces and assess the impact of various clustering algorithms and distance measures.

II. Related Work

The use of playtraces has a long history in the field of videogames. Examples include the analysis of playtraces to identify player styles [1] and to understand player behavior [2]. In addition, Campbell et al. [3] investigated a variety of dissimilarity metrics and clustering algorithms to understand player activity in some example levels from stealth and platform games.

Construction of playtraces for arbitrary games was discussed in [4]. Here, playtraces are developed from in-game actions and a distance metric based on the edit distance is used to group similar playtraces together, in order to visualize strategies. However, the examples given relate only to videogames.

The use of playtraces and playtrace clustering within the context of board and card games has been much more limited. To the best of the authors knowledge, only Bendekgey [5] has specifically investigated the clustering of playtraces within a modern tabletop game. They investigate playtraces generated from human play of *Dominion* obtained from an online game server. Clusters are visually identified using t-distributed stochastic neighbor embedding, with the resulting visualizations qualitatively examined for player strategies.

Received 2 August 2024; revised 18 November 2024; accepted 18 December 2024. Date of publication 23 December 2024; date of current version 17 September 2025. Recommended by Associate Editor S. Cooper.

Anthony Owen is an independent researcher. He resides in W5 3UR London, UK (e-mail: anthonyowen1@gmail.com).

Digital Object Identifier 10.1109/TG.2024.3520862

Here, we use AI simulated playtraces to allow a greater variety of game configurations to be analyzed. We provide an exploration of alternative clustering methods and distance measures with a quantitative measure of the strength of the clustering. In addition to playtraces based on card counts, we examine sequences of player actions in the form of N-Grams. We also demonstrate visualisation of playtraces for interpretation by game designers and easier identification of card synergies.

The use of AI-based simulated playtesting of games has also been applied to analyzing game balance. The literature related to automated game balancing of DBCGs is limited to three examples, two of which utilise *Dominion* as a test case. These three examples consist first of [6], which utilizes genetic algorithms to assess game balance in *Dominion* with respect to the availability of specific game mechanics (represented by a common set of cards that players can obtain during game play). Second, in [7] Monte Carlo tree search (MCTS) is used to run repeated game sessions evaluating the influence of particular cards and pairs of cards on the outcome. Finally, Budijono et al. [8] discussed an auto-battler card game developed for the express purpose of AI research, and proposes a number of metrics that could be used to assess meta-game health in deck building card games.

This article contributes to the existing literature by demonstrating how both AI simulated game sessions and human play of a DBCG can be used to generate playtraces, which when clustered can aid in identification of strategies and card synergies.

III. Background

A. Dominion

Dominion is a DBCG by Rio Grande Games, designed by Donald Vaccarino. Below is a brief overview of gameplay; full rules are available in the rulebook.[1]

The game features seven base cards: four victory point (VP) cards (Curse: -1VP, Estate: 1VP, Duchy: 3VP, and Province: 6VP) and three treasure cards (Copper: 1, Silver: 2, and Gold: 3). In addition, a Kingdom card set includes ten chosen action card types, each with ten copies. These cards influence game mechanics and strategies. Cards are purchased using treasure, and VP cards have a limited supply based on player count.

Each player begins with a starting deck consisting of seven copper and three Estate cards, which is shuffled to form a draw pile. At the start of each round, players draw five cards. Turns consist of three phases: action, buy, and clean-up.

1) *Action phase:* The player plays an action card and resolves its effects. Cards offering "+1 Action" allow additional plays. The phase ends when no actions remain.
2) *Buy phase:* Treasure from the action phase is used to buy one or more cards from the Kingdom or base sets ("+1 Buys" can be obtained in the action phase).
3) *Clean-up phase:* Played and purchased cards go to the discard pile, and the player draws five new cards. If the draw pile is empty, the discard pile is shuffled to form a new one.

[1] [Online]. Available: www.riograndegames.com/wp-content/uploads/2016/09/Dominion2nd.pdf

The game ends when either the Province pile or three Kingdom card piles are empty. Players sum the VP in their decks, and the highest total wins.

IV. Playtrace Definitions and Distance Measures

A. Card Count Playtraces and L_k-Norm

A card count playtrace represents the composition of a player's deck at the end of each round. For player p in game n at the end of round t, we denote a deck vector by $\vec{d}_n^{p,t}$. This is a vector of length M, the number of distinct card types available in the supply. Each entry $d_{n,i}^{p,t} \in \mathbb{Z}_0^+$, indicates the count of cards of type i in a players deck (including draw, hand, discard and played card piles). A card count playtrace, $\vec{T}_n^p$, is then defined by

$$\vec{T}_n^p = (\vec{d}_n^{p,1}, \vec{d}_n^{p,2}, \ldots, \vec{d}_n^{p,R}) \tag{1}$$

where R is the number of rounds.

A playtrace, represented as a vector of integer vectors, can be flattened into a single integer vector of length MR. We can define a distance or dissimilarity metric between two playtraces utilizing the L_k-norm

$$L_k(\vec{T}_{n_1}^{p_1}, \vec{T}_{n_2}^{p_2}) = \left(\sum_{t=1}^{R} \sum_{i=1}^{M} (d_{n_1,i}^{p_1,t} - d_{n_2,i}^{p_2,t})^k \right)^{\frac{1}{k}}. \tag{2}$$

Note that for k equal to two we have the usual Euclidean distance measure. Also, we can normalize a playtrace vector $\vec{T}_n^p$ by diving each entry by $L^k(\vec{T}_n^p, \vec{0})$, where $\vec{0}$ is a vector of zeros with the same length as $\vec{T}_n^p$.

Since games of *Dominion* vary in length, R is set to the maximum number of rounds within a game session. When a game ends, deck vectors are frozen and repeated for subsequent rounds up to R, ensuring all playtraces have the same length for distance measurement.

In our case, given that R can be of the order of 50, playtraces will have roughly 850 dimensions (M is 17 in *Dominion*). Fractional L_k-norms (k less than one), as suggested by Aggarwal et al. [9], can improve discrimination in higher dimensions. Thus, we also explore the impact of varying k in the formation of our playtrace clusters.

B. N-Grams and Jensen–Shannon Distance

An alternative approach to using the game state space is to use the action space. We define a string, S_n^p, representing the list of actions taken by player p in game n. For example

PlayWitch BuySilver PlayWorkshop GainGardens PlayChapel TrashCopper BuyProvince....

We convert this string into a playtrace, $S_n^p(N)$, by mapping it to a multiset of N-Grams. N-Grams, drawn from natural language processing [10], are sequences of N words or in our case, actions. For example, with bi-grams, each action is paired with neighbouring actions to generate a multiset of action pairs

{(PlayWitch, BuySilver), (BuySilver, PlayWorkshop), (PlayWorkshop, GainGardens), (GainGardens, PlayChapel), (PlayChapel, TrashCopper)... }.

A uni-gram-based playtrace is a multiset containing individual actions, whereas for higher order N-Grams our playtrace consists of a multiset containing all observed sequences of N actions taken in order.

To simplify playtraces and improve interpretability, we make two assumptions. First, we introduce an aggregated action, "end current phase," used when a player ends their turn or plays only treasure cards without buying or playing Kingdom cards. Second, we exclude actions taken in response to opponents, such as discarding cards after an opponent plays Militia. These adjustments focus the analysis on individual player strategies and helps with interpretability of the playtraces.

To measure the distance between N-Gram playtraces, we use the Jensen–Shannon measure, which quantifies dissimilarity between probability distributions. Each playtrace $S_n^p(N)$ is mapped to a corresponding discrete probability distribution, $SP_n^p(N)$ over the set $V(N)$, of all N-grams observed in our game session. For each $G_i \in V(N)$, we count its' occurrences in $S_n^p(N)$, setting unobserved counts to zero, and normalize to compute probabilities.

We can now calculate our Jensen–Shannon distance measure between two playtraces, $S_{n_1}^{p_1}(N)$ and $S_{n_2}^{p_2}(N)$. To simplify notation we define $p = SP_{n_1}^{p_1}(N)$, $q = SP_{n_2}^{p_2}(N)$ and $\mathrm{m} = \frac{1}{2}(p+q)$ which is the average of our two probability distributions. In addition p_i and q_i denote the probabilities of N-Gram i in p and q, respectively. Then, we have

$$JS(S_{n_1}^{p_1}(N), S_{n_2}^{p_2}(N)) = \sqrt{\frac{1}{2}(KL(p,m) + KL(q,m))} \quad (3)$$

where

$$KL(p,q) = \sum_{i=1}^{|V(N)|} f(p_i, q_i). \quad (4)$$

Here, $f(p_i, q_i) = p_i log(\frac{p_i}{q_i})$, if both p_i and q_i are nonzero, otherwise it is zero. Note that the Jensen–Shannon distance measure has the benefit of accommodating playtraces of variable length.

V. Methods

A. Clustering Algorithms and Visualization

Now that we have our playtraces and associated distance measures, we can explore how we can cluster playtraces into groups of similar playtraces. First, we use K-Means [11], which is simple and fast, but requires selecting the number of clusters, K, in advance, and assumes spherically distributed clusters. In addition, K-Means requires a Euclidean distance measure to guarantee convergence. This can be a disadvantage, since the discriminatory power of the Euclidean distance deteriorates in higher dimensions [9].

Next, we use K-Medoids [12], a variant of K-Means, where cluster centroids are selected from the data rather than computed as averages. This is useful here, as averaging playtraces may suggest card combinations that never appear together. K-Medoids also works with non-Euclidean distance measures.

Finally we use density-based spatial clustering of applications with noise (DBSCAN). DBSCAN has two hyperparameters, m and ε, that define the minimum number of neighboring points for a point to be considered a core point, and the size of that neighborhood respectively. Core points are identified to form clusters, as discussed in [13]. Benefits of this approach include identifying nonspherical clusters, assigning data points as noise as opposed to forcing a cluster assignment and supporting non-Euclidean distance measures.

In this case, the concept of a cluster centroid can be harder to define. For simplicity, we will define it as the nearest playtrace to the mean playtrace for a cluster. This may not always be representative of the cluster, especially if it is not spherically distributed.

We visualize the clusters using metric multidimensional scaling (MDS) [14], which takes a distance matrix and reconstructs coordinates in a lower dimensional space to closely preserve the original distances, enabling cluster visualization.

We can also use silhouette averages (SA) [15] to assess cluster quality, this is calculated as the average of the silhouette coefficients over all playtraces. The silhouette coefficient for a data point (in a cluster of size greater than one) is defined by

$$s_i = \frac{b_i - a_i}{\max\{a_i, b_i\}}. \quad (5)$$

Here, a_i is the mean distance between playtrace, i, and all the other playtraces within its cluster. Conversely, b_i, is found by calculating for each cluster the average distance between playtrace i and the playtraces in that cluster (not including the cluster playtrace i belongs to), then taking the smallest of those values. These terms can be expressed as

$$a_i = \frac{1}{|C_I| - 1} \sum_{j \in C_I, i \neq j} \mathrm{dist}(i,j) \quad (6)$$

and

$$b_i = \min_{J \neq I} \frac{1}{|C_J|} \sum_{j \in C_j} \mathrm{dist}(i,j) \quad (7)$$

where $\mathrm{dist}(i,j)$ could be either the L_k or Jensen–Shannon norms.

A SA close to one indicates playtraces are well-clustered and distant from neighboring clusters. An average near minus one suggests playtraces are similar to those in adjacent clusters.

A commonly accepted rule of thumb is that a SA greater than 0.25 indicates the presence of some degree of clustering. However, there is no theoretical basis for choosing a threshold value for the SA, additionally it can be hard to achieve higher values for high dimensional data. As such clusters should be evaluated with human inspection [15]. This is of particular importance in our use case, where playtraces and playtrace clustering can support game designers in identifying strategies and card synergies, augmenting more traditional approaches like human-based playtesting.

B. Tabletop Games Framework (TAG)

In order to generate our playtraces we use an implementation of *Dominion* contained in the TAG [16]. This framework also provides the MCTS algorithm (see [17] for an introduction to

TABLE I
KINGDOM CARD SET—SD

Name	Effect Summary	Cost
Chapel	Trash up to four cards from your hand	3
Workshop	Gain a card costing up to 4	3
Throne Room	Play an action card in hand twice	4
Bureaucrat	Gain a Silver, opponents redraw VP cards	4
Gardens	Worth 1VP for every 10 cards in player's deck	4
Bandit	Gain a Gold, opponents trash 1 treasure card	5
Festival	+2 Actions, +1 Buy, +2 treasure	5
Witch	+2 Cards, opponents gain a Curse	5
Sentry	+1 Card, +1 Action, trash 2 cards	5
Artisan	Gain a card costing up to 5	6

MCTS), finite state machine (FCM) agents and the tournament functionality used in this article.

The purpose of TAG is to promote research into the uses of AI in modern tabletop games. It provides a common API for games and game playing agents, allowing for a comparison of AI methods across a broad spectrum of tabletop games. It is used to support cross board game analyses and also to explore reinforcement learning in board games [18]. The game engine for *Dominion* in TAG is well supported and has been used in other research, for example [19] and [20].

C. FCM Agents

We use two FSM to simulate two well known strategies specific to the "size distortion" (SD) Kingdom card set described in Table I (see the *Dominion strategy wiki*[2] for a full description of card effects). This Kingdom card set is recommended in the base game of *Dominion*.

Our first FSM agent plays a Big Money with Gardens strategy (BMWG) with the SD set of cards. Big Money is a base case strategy in *Dominion*, and focuses on buying treasure cards that in turn allow the purchase of Province cards for VPs. Big Money with Gardens is a variant of the Big Money strategy, adding the purchasing of Gardens to generate additional VPs.

Our second FSM agent implements the Double Witch (DW) strategy. This strategy aims at using the Witch card to flood an opponent's deck with Curse cards. This not only reduces their VP score, but also reduces the probability of the opponent drawing useful cards.[3]

We will use FSM agents primarily as a base case, and a means of demonstrating how distinct strategies can result in distinct clusters within our playtrace dataset.

We have chosen these two FSM agents as they represent two distinct strategies, one involving a focus on generating treasure and the other on negatively impacting an opponent's deck. Other FSM agents could also be chosen, in particular Provincial AI [21] effectively trains parametrised FSM agents on individual Kingdom card sets. This could be used to generate other FSM agents to further explore playtrace clustering on a variety of Kingdom card sets.

[2] [Online]. Available: https://wiki.dominionstrategy.com/

[3] [Online]. Available: Code for both our FSM agents can be found at https://github.com/owenant/TabletopGames_ForDominionPlayTraces

TABLE II
AVERAGE WIN RATES OVER 100 GAMES WITH THE SD KINGDOM CARD SET (HIGHEST WIN RATES IN BOLD)

	MCTS	BMWG	DW
MCTS	53%	**83%**	40%
BMWG	**17%**	50%	1%
DW	60%	**99%**	50%

D. MCTS Agent

In addition to our FSM agents, we also use an MCTS-based agent. We use a single MCTS agent that combines both Information Set MCTS (ISMCTS) [22] and multitree MCTS [23]. In ISMCTS, each node in the tree is defined by a set of actions from the root node, as opposed to being a specific game state at each node. In addition, hidden information is resampled at the root node for each simulation loop. In multitree MCTS, a tree is constructed independently for each player

ISMCTS helps with the hidden information aspects of *Dominion* and multitree MCTS trades off greater search depth at the expense of reacting to opponent's moves. Due to the indirect nature of the interaction between players in *Dominion*, this tradeoff is expected to be positive.

Our MCTS agent will also use the following features.

1) UCT-tuned [24] as our tree policy in the selection phase (with $K = 1$).
2) The simulation play-outs are performed to a length of 30 actions, with a playout policy determined by a simple score-based heuristic (actions are chosen to maximise the player's score for the next turn).
3) The playouts also use the moving average sampling technique (MAST) [25], this determines the average reward from an action independent of its position in the game tree, and biases future decisions based on those rewards. We set the MAST gamma value to 0.5, with an explore epsilon of 0.1.
4) A time budget of 500 ms will be allotted for each move.

Note that our choice of heuristic is inline with empirical evidence on the performance of MCTS in *Dominion* given in [20]. However, we found that a roll-out length of 30 combined with MAST gives a superior performance to a longer roll-out with a random play-out policy (57% win rate).

We compare win rates between our MCTS agent and FSM agents using the SD Kingdom card set. The average win rates are given in Table II, with each win rate given as the win rate for the AI agent in the row versus the AI agent in the column.

Note that the DW agent is the strongest, however this strategy is specific to the SD card set, and the MCTS agent is capable of playing any Kingdom card set.

VI. EXPERIMENTS

We perform a number of experiments using both FSM and MCTS agents competing in round robin tournaments of 500 games, with each bot switching between being first and second player after 250 games. These tournaments are run using two different Kingdom card sets in TAG. Data on player actions and the game state space is recorded at the end of each round,

TABLE III
SA WITH A TOURNAMENT OF 500 GAMES FOR BMWG VERSUS DW USING THE SD CARD SET

Playtrace	Algorithm	Norm	SA	Hyperparams
Card count	K-Means	L_2	0.59	$K = 2$
Card count	K-Medoids	L_2	0.59	$K = 2$
Card count	K-Medoids	$L_{0.5}$	**0.80**	$K = 2$
Card count	DBSCAN	L_2	0.58	$m = 5, \varepsilon = 0.16$
Card count	DBSCAN	$L_{0.5}$	**0.74**	$m = 5, \varepsilon = 0.14$
Uni-gram	K-Medoids	JS	0.67	$K = 2$
Bi-gram	K-Medoids	JS	0.52	$K = 2$
Tri-gram	K-Medoids	JS	0.38	$K = 2$
Uni-gram	DBSCAN	JS	0.67	$m = 5, \varepsilon = 0.11$
Bi-gram	DBSCAN	JS	0.52	$m = 5, \varepsilon = 0.26$
Tri-gram	DBSCAN	JS	0.38	$m = 95, \varepsilon = 0.46$

Bold values indicate the most prominent clustering.

and then processed to form our card count and N-Gram-based playtraces.

In order to determine the optimal hyperparameters for our various clustering methods, we perform a grid search maximising the SA. For the grid search for K-Means and K-Medoids we let K vary between 2 and 5 clusters. For DBSCAN, we vary m from 5 to 250 in steps of size 10, and ε from 0.01 to 0.995 in steps of 0.005. The maximum value for m is based on our data sample size (we will run 500 games for MCTS). In addition, we normalize our card count playtraces so that the maximum distance between any two playtraces is two. We set a maximum ε as half that maximum value.

A. FSM Agents With SD Kingdom Card Set

We start by establishing a baseline set of results using the two FSM agents corresponding to BMWG and DW strategies. We run a tournament consisting of 500 games using the SD kingdom card set. Optimal SA from our grid search (along with the corresponding hyperparameters) for our various playtrace types, distance measures and clustering algorithms are shown in Table III.

We see that in the card count case the $L_{0.5}$ norm gives the highest SA, inline with our expectations that in a high dimensional space the discriminatory power of the Euclidean norm is reduced. We also see that the uni-gram result is slightly higher than the L_2 card count case, however we see a deterioration in SA as we move from uni- to tri-grams. For uni-, bi- and tri-grams there are 78, 870 and 3694 distinct N-Grams observed, respectively, in our tournament. As the order of the N-Grams increases and due to the relatively short length of a *Dominion* game, it becomes progressively more unlikely that playtraces will share common sequences of actions, which leads to an average overall increase in distance between playtraces and a diffusion of clusters.

Another observation is that our DBSCAN and K-Medoids results are very close, suggesting that our clusters are roughly spherical. The DBSCAN and K-Medoids MDS plots are virtually identical in this case, and therefore we only show some example DBSCAN MDS plots in Fig. 1.

The corresponding DBSCAN centroids for the card count case with $L_{0.5}$-norm are shown in Fig. 2. Note that when displaying cluster centroids for card count playtraces, we suppress card

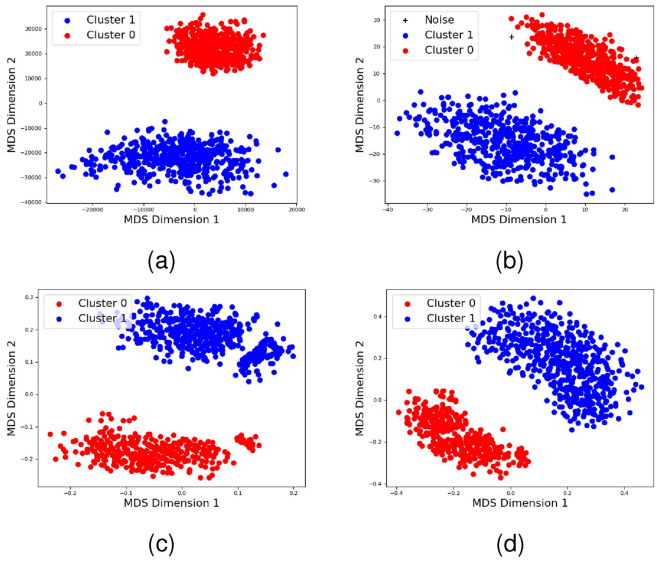

Fig. 1. DBSCAN MDS plots for BMWG versus DW agents. (a) Card count with $L_{0.5}$. (b) Card count with L_2. (c) Uni-grams. (d) Bi-grams.

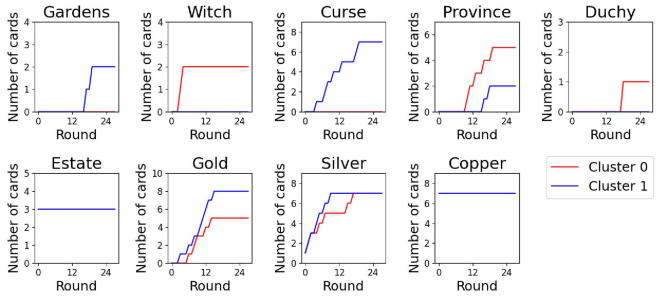

Fig. 2. DBSCAN centroids for BMWG versus DW agents with card count playtraces and $L_{0.5}$-norm.

types that are not contained in the deck of either agent. Cluster zero clearly shows a DW strategy, since the Witch card count reaches a maximum of two, and there is a focus on Silver and Gold to purchase Province and Duchy VP cards. Cluster one shows a BMWG strategy, which can be observed by the focus on treasure cards and the purchase of Gardens cards for additional VPs. Note that the purchase of Province cards is somewhat limited, this is driven by the influx of curse cards from the DW opponent, weakening the strength of the agent's deck.

Examining the centroids in the bi-gram case in Fig. 3, we again see our two strategies reflected in the clusters. Cluster one reflects a BMWG strategy and has a high frequency of bi-grams containing a buy Gold or buy Silver action, reflecting it's focus on treasure cards. In addition, we see purchasing of Garden cards for additional VPs. Cluster zero follows a DW strategy and has a number of bi-grams containing either a play or buy Witch action. The remaining bi-grams for cluster zero do not contain any other Kingdom cards, but solely focus on treasure cards and VP cards.

This demonstrates that clustering of card count and N-Gram playtraces is capable of separating out different strategies, at least in this simplified case with two players playing separate and clearly defined strategies. More generally, we would expect

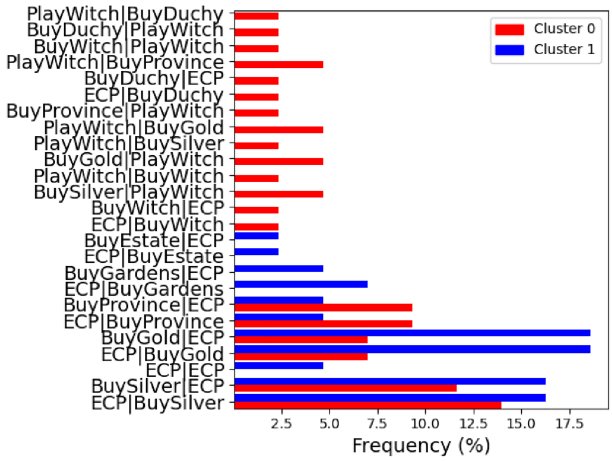

Fig. 3. DBSCAN centroids for BMWG versus DW agents for bi-grams.

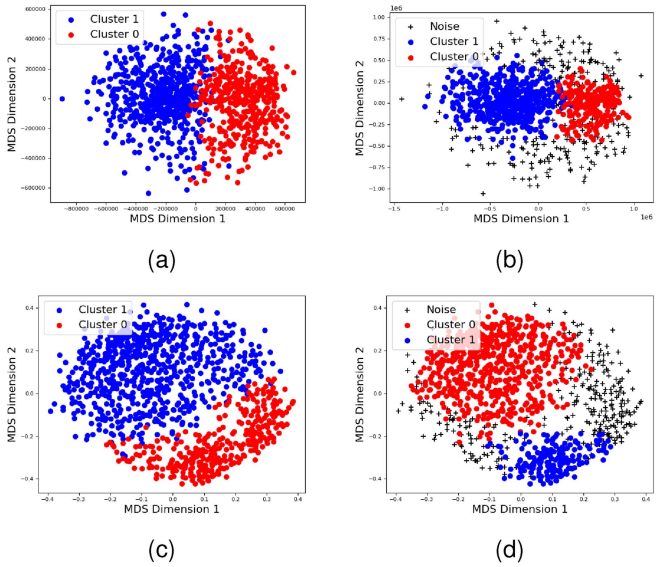

Fig. 4. MDS plots for MCTS agents. (a) K-Medoids with card count and $L_{0.5}$. (b) DBSCAN with card count and $L_{0.5}$. (c) K-Medoids with uni-grams. (d) DBSCAN with uni-grams.

to see distinct clustering between playtraces generated by two FSM agents as long as the deterministic choices made by each agent are sufficiently different from the other.

B. MCTS With SD Kingdom Card Set

Next we move onto the case of MCTS, where we do not know the strategy being played upfront. We expect our MCTS agents to provide strong enough play to utilize card synergies, which are highlighted in the generated playtraces. Across the games played we would also expect variation in play due to the stochastic nature of the game environment and the MCTS algorithms (which utilize different seeds) combined with the basic structure of DBCGs. In DBCGs optimal actions are a function of current deck composition, which is in turn a result of all previous actions. So variations in actions at the start of the game can lead to a large divergence in deck composition by the end of a game. This is compounded by the initial card draw on the first round, which can result in two to five coppers being drawn, greatly changing the range of options for card purchase on the first turn. In section VI-D we demonstrate how restricted play can also improve the variation in the play of an MCTS agent to further explore the strategy space.

We again use the SD Kingdom card set and run a tournament consisting of 500 games. We remove outlier games with a length greater than 55 rounds or with a score higher than 100, which account for 1% of our playtraces. SA are shown in Table IV.

We see lower SA as compared to the FSM case, since both players are simulated using the same MCTS algorithm as opposed to two different bots with distinct strategies. Except for the bi- and tri-gram cases, the SA are strong enough to indicate the presence of clusters, and the SA for K-Means and K-Medoids are maximised with K equal to two. The presence of two clusters can also be confirmed through visualisation using the MDS plots in Fig. 4. We see two clusters that are clearer in the DBSCAN case where data points can be assigned as noise as opposed to being forced into a specific cluster.

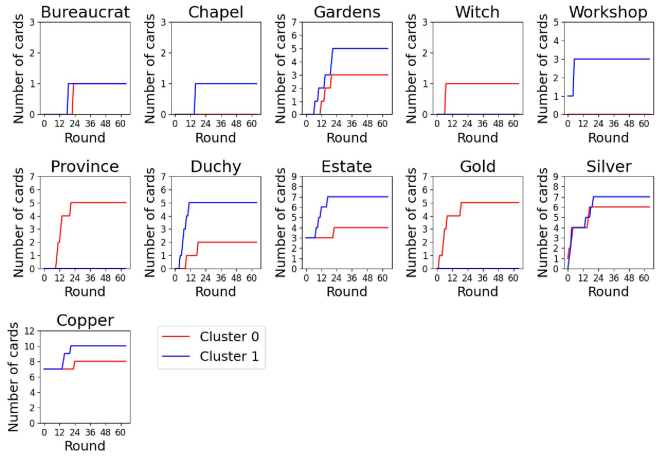

Fig. 5. DBSCAN centroids for MCTS agents with card count playtraces and $L_{0.5}$-norm.

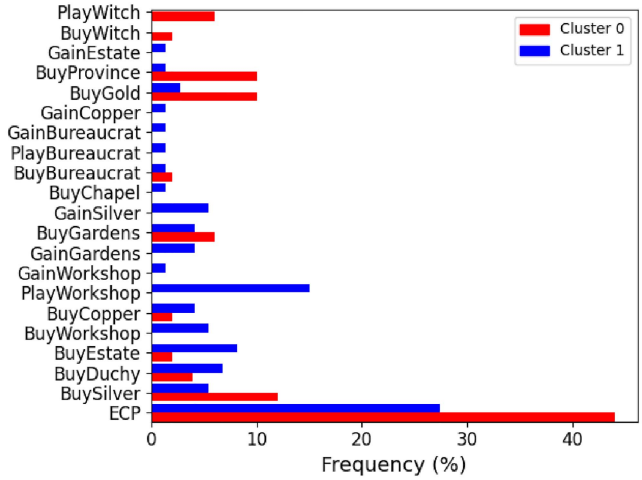

Fig. 6. DBSCAN centroids for MCTS agents for uni-grams.

TABLE IV
SA WITH A TOURNAMENT OF 500 GAMES BETWEEN MCTS AGENTS USING THE SD CARD SET

Playtrace	Algorithm	Norm	SA	Hyperparams
Card count	K-Means	L_2	0.27	$K = 2$
Card count	K-Medoids	L_2	0.27	$K = 2$
Card count	K-Medoids	$L_{0.5}$	**0.35**	$K = 2$
Card count	DBSCAN	L_2	0.35	$m = 115, \varepsilon = 0.315$
Card count	DBSCAN	$L_{0.5}$	**0.4**	$m = 95, \varepsilon = 0.37$
Uni-gram	K-Medoids	JS	0.26	$K = 2$
Bi-gram	K-Medoids	JS	0.15	$K = 2$
Tri-gram	K-Medoids	JS	0.07	$K = 2$
Uni-gram	DBSCAN	JS	**0.35**	$m = 135, \varepsilon = 0.26$
Bi-gram	DBSCAN	JS	0.21	$m = 185, \varepsilon = 0.44$
Tri-gram	DBSCAN	JS	0.11	$m = 95, \varepsilon = 0.6$

Bold values indicate the most prominent clustering.

We again examine our cluster centroids shown in Figs. 5 and 6. The cluster centroids for the DBSCAN case are similar to the equivalent K-Medoids case.

The two clusters show distinct strategies. The cluster one centroid displays a well known "Gardens/Workshop" combination focus. This is shown in the card count case by these being the main Kingdom cards purchased, and in the uni-gram case by the relatively high frequency of play Workshop, and also buy and gain Gardens action. Garden cards cost 4 and are worth 1VP for each 10 cards in the player's deck. The Workshop card allows a player to pick up any card with a cost of 4 or less for free. In this case, the strategy is to use the Workshop cards to gain Garden cards (in addition to purchasing them directly), generating VPs.

The cluster zero centroid has a strong focus on treasure cards, particularly Gold and also purchasing Province cards and some Garden cards for VPs. This is a BMWG. We also see some use of the Witch card. Note that we do not include gain curse in our N-Grams actions as it is a response to another player's action, and those actions are ignored to simplify our N-Grams. The lack of curse cards in the centroids for the card count case, seems to suggest that the Witch card is not played enough by our MCTS agent to generate enough curse cards that they appear in the cluster centroids.

In addition, we note that there is limited use of the chapel card. This is commonly known to be a strong card in *Dominion* that allows a player to trash or remove weaker cards from their deck, thereby increasing the overall strength of their deck. The MCTS agent does not tend to make use of this card, this is most likely due to the impact of trashing requiring a relatively deep exploration of the game tree for the benefits to become apparent. These examples show that not all available strategies are captured by our MCTS agent.

C. Comparison of Human and MCTS Generated Playtraces With the FG1E Kingdom Card Set

Using MCTS generated playtraces allows us to generate a large amount of data quickly, however human playtesting is essential. In this section, we compare playtraces generated by MCTS to those generated by human play.

In order to do this, we examine player logs collected by dominion.isotropic.org between October 2010 and March 2013. The most popular Kingdom card set is the First Game 1st Edition (FG1E) set, a summary of this Kingdom card set is given in Table V.

TABLE V
KINGDOM CARD SET—FG1E

Name	Effect Summary	Cost
Cellar	+1 Action, discard x cards then draw x cards	2
Moat	+2 Cards, reveal to protect from attack	2
Village	+1 Card, +2 Actions	3
Woodcutter	+1 Buy, +2 treasure	3
Workshop	Gain a card costing up to 4	3
Smithy	+3 Cards	4
Remodel	Trash a card and gain a card worth 2 more	4
Militia	+2 treasure, opponents discard down to 3 cards	4
Market	+1 Card, +1 Action, +1 Buy, +1 treasure	5
Mine	Trash a treasure, gain a treasure worth 3 more	5

TABLE VI
SA FOR MCTS AGENTS AND HUMAN PLAY USING THE FG1E CARD SET

Playtrace	Algorithm	Norm	SA-MCTS	SA-Human
Card count	K-Means	L_2	0.17	0.15
Card count	K-Medoids	L_2	0.14	0.10
Card count	K-Medoids	$L_{0.5}$	0.16	0.12
Card count	DBSCAN	L_2	0.34	N/A
Card count	DBSCAN	$L_{0.5}$	0.57	N/A
Uni-gram	K-Medoids	JS	0.30	0.08
Bi-gram	K-Medoids	JS	0.06	0.10
Tri-gram	K-Medoids	JS	0.02	0.04
Uni-gram	DBSCAN	JS	0.20	0.39
Bi-gram	DBSCAN	JS	0.24	N/A
Tri-gram	DBSCAN	JS	N/A	N/A

Restricting ourselves to two player games we find 149 relevant log files. The actions of each player for each round is parsed and then converted into card count and N-Gram-based playtraces. Then, in-conjunction with an MCTS tournament of 500 games played with the FG1E kingdom card set, we can compare clustering between AI and human play. We start by examining the SA in Table VI .

In the cases where we have N/A the algorithm is assigning all playtraces to a single cluster, and hence we can not compute the SA. The nonzero DBSCAN SA are also somewhat misleading, as we generally have one large cluster with a few remaining playtraces assigned to a second cluster. Examples are shown in Fig. 7. The only case where we see clustering is in the MCTS uni-gram case for K-Medoids, shown in Fig. 7(d).

Fig. 8 shows the cluster centroids in the K-Medoids uni-gram case. The centroids show similar strategies. Cluster one is showing a Smithy plus Big Money strategy, which focuses on collecting treasure cards, such as Silver and Gold in order to buy Provinces, and then speeds up this process by using Smithy to draw extra treasure cards. Cluster zero is similar, except makes more use of the moat card as opposed to the Smithy, which allows for a smaller number of new card draws but is also cheaper.

Given the lack of observable clustering for this Kingdom card set, we compute an average playtrace across all available playtraces. Despite the lack of clustering, this can still provide useful information as to the most popular cards played within a given Kingdom card set, for both our MCTS and human players, providing valuable information to a game designers. In Fig. 9, we have the average card count playtraces for both human and MCTS agents, and in Fig. 10 the corresponding uni-gram case (note we have removed uni-grams with a frequency of less than 1.5% for clarity).

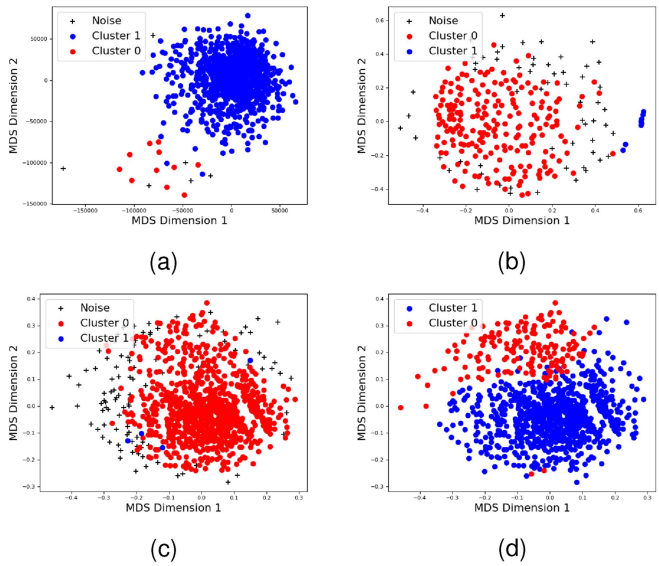

Fig. 7. MDS plots for FG1E card set. (a) DBSCAN for MCTS with card counts and $L_{0.5}$-norm. (b) DBSCAN for human players with uni-grams. (c) DBSCAN for MCTS agent with uni-grams. (d) K-Medoids for MCTS with uni-grams.

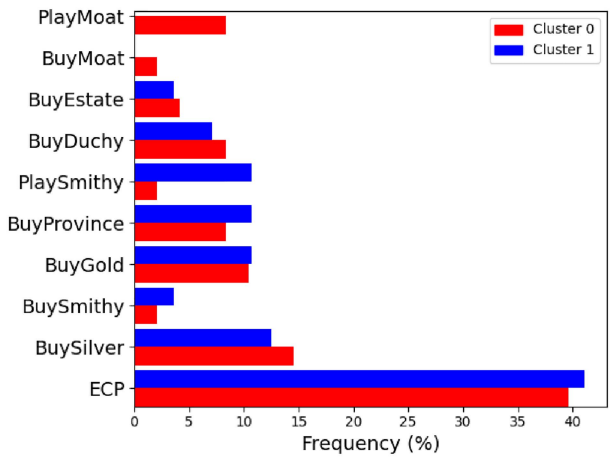

Fig. 8. Cluster centroids for K-Medoids uni-gram clustering of MCTS generated playtraces for FG1E card set.

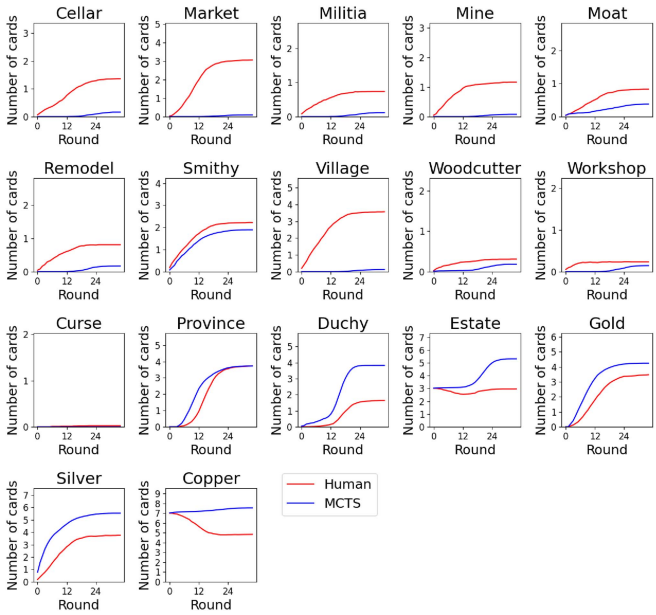

Fig. 9. Mean card count playtraces with $L_{0.5}$-norm for human players and MCTS agents for FG1E card set.

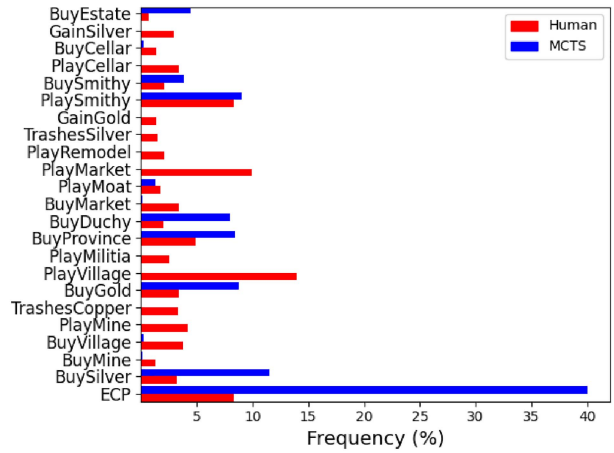

Fig. 10. Mean uni-gram probability distribution for human players and MCTS agents for FG1E card set.

Here, we see more evidence that the MCTS agent is generally playing a Smithy plus Big Money strategy. This is clear from the average card count playtrace where we see the main action card being purchased by the MCTS agent is Smithy. In addition, the uni-gram playtrace shows a focus on playing Smithy, buying Gold and Silver and also the Province and Duchy VP cards.

For human play, we see more varied play as compared to the MCTS case. In particular, we see much more use of village and market cards, in-conjunction with Smithy. This reflects a more advanced Smithy/Village engine-based strategy. In this strategy, the aim is to cycle through the entire deck each round, with the Village providing additional cards and actions that allow a subsequent play of a Smithy. Ideally the treasure in a player's deck should then be enough to buy a Province or Duchy card each time. Typically a market card is also added to provide additional buy actions so that more than one victory card can be bought at a time.

Another difference between our MCTS and human-based playtraces is the latter's use of the mine card. This allows a player to trash copper cards, which are weak. This increases a player's probability of drawing higher value treasure cards and is a common technique in *Dominion* for strengthening a deck. This tactic was not obviously exhibited in the MCTS case.

D. Using Restricted Play to Increase Variation in MCTS Playstyle

As we have seen in previous sections, our MCTS agent does not use all available strategies and the generated playtraces provide only a restricted view on card synergies. To partially alleviate the impact of this problem, we can use the restricted play framework proposed in [26]. In this framework, the impact

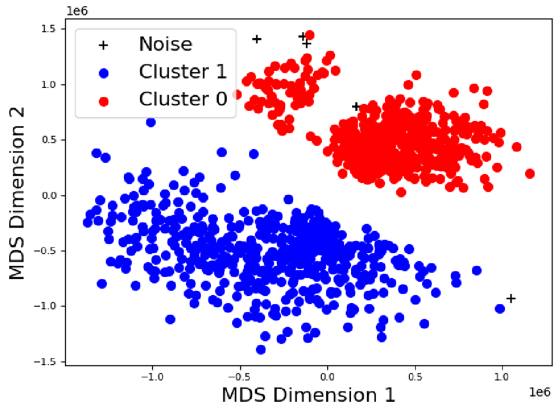

Fig. 11. MDS DBSCAN plot for cardcount playtraces with $L_{0.5}$-norm and MCTS restricted play with SD Kingdom card set.

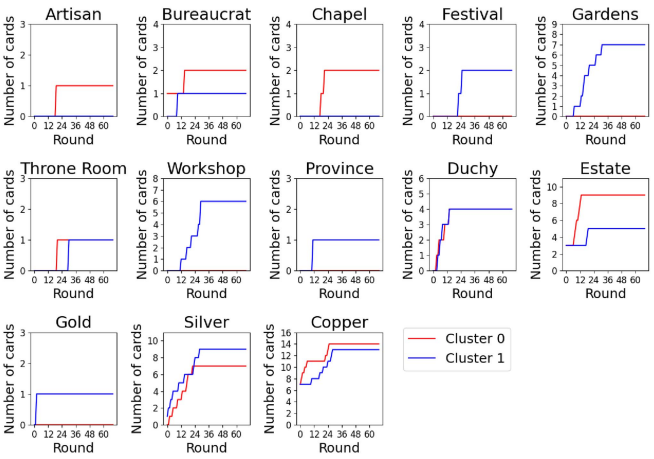

Fig. 12. DBSCAN cluster centroids for cardcount playtraces with $L_{0.5}$-norm and MCTS restricted play with SD Kingdom card set.

of a game dynamic or component is assessed by examining the impact of removing the access of a player to that component. In our case, we prevent our MCTS agent from purchasing particular cards, which forces our agent to use alternative card combinations and strategies, which are then highlighted via playtraces.

We use the same set-up as in Section VI-B, with the exception that a single MCTS bot is restricted from buying Gardens, Workshop, Gold, Silver and Province cards. This prevents the bot from exploring the Gardens/Workshop and BMWG strategies already identified. Focusing on card count playtraces with an $L_{0.5}$ norm, we have the DBSCAN MDS plot shown in Fig. 11. The average silhouette score is 0.45, and shows two distinct clusters.

Cluster one contains playtraces generated by the unrestricted MCTS bot, and cluster zero playtraces generated by the restricted MCTS bot. Examining the corresponding centroids in Fig. 12, we see that the restricted MCTS bot uses a combination of Artisan and Bureaucrat to obtain Silver cards, which are then used to purchase Estate cards for VPs. Artisan allows the player to gain a card of cost up to five and Bureaucrat directly adds a Silver card to the top of the player's draw deck. Whilst

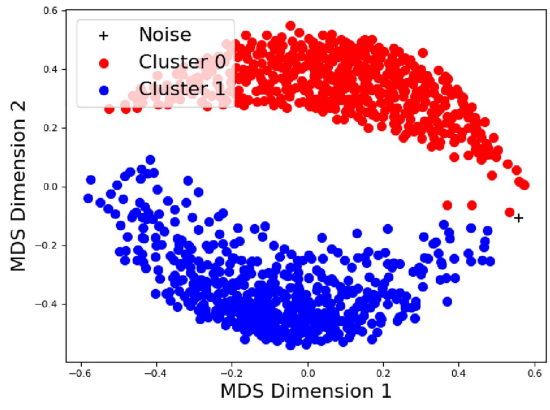

Fig. 13. MDS DBSCAN plot for bi-grams and MCTS restricted play with FG1E Kingdom card set.

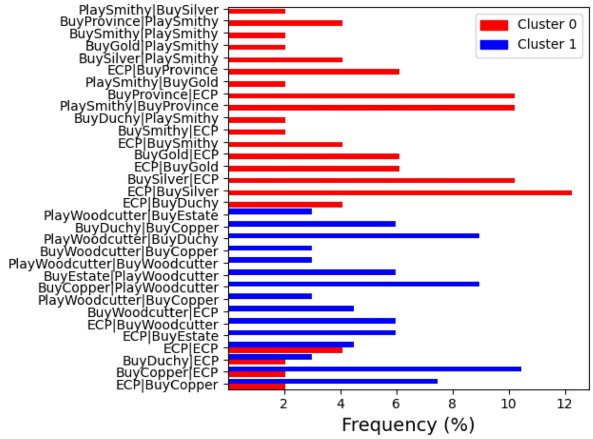

Fig. 14. DBSCAN bi-gram cluster centroids for MCTS restricted play with FG1E Kingdom card set.

a suboptimal strategy, it does highlight how a player can use Kingdom cards to increase their stock of treasure cards.

As an additional example, we perform a similar analysis with the FG1E Kingdom card set, restricting a single MCTS bot from using Smithy, Gold, Silver and Province cards. We focus on bi-grams and the MDS plot for DSCAN clustering is shown in Fig. 13. The two clusters have an average silhouette value of 0.42. Examining the centroids in Fig. 14, we see that cluster zero (corresponding to unrestricted MCTS playtraces) follows a Smithy plus Big Money strategy. However, for cluster one we see substantial use of the Woodcutter card. This card gives two treasure and plus one buy action, and can be an alternative source of treasure to Silver cards supporting the purchasing of the Duchy and Estate VP cards.

VII. Challenges and Opportunities

A. Investigating Game Phase Dependent Strategies

Here, we have examined playtraces for full games, however it would also be useful to examine how strategy and card usage changes between early, mid, and late game phases. For example, in most DBCGs the early game is characterized by building an engine and the later game phase by the acquisition of VPs.

Playtrace clustering on game phases could help highlight how play changes as a game progresses.

B. Impact of AI Agent on Strategy Identification

One potential avenue for future research is the impact the choice of AI agent can have on our clustering results and hence strategy identification. Strategies can only be discovered if the AI agent plays those strategies. As a simple example, if we move from a score-based heuristic to a simple win only heuristic for our MCTS playout policy, we see more use of the trashing mechanic and also use of Witch cards. Another possibility is to compare playtraces generated by MCTS with other AI algorithms, such as deep reinforcement learning-based approaches [27].

C. Investigation of Other Kingdom Card Set Choices

We saw distinct clustering in the SD Kingdom card set, however both MCTS and human generated playtraces did not show distinct clustering for the FG1E set. This could be for a number of different reasons. First, the SD Kingdom card set tends to have more distinct strategies, e.g., BMWG versus DW, whereas the FG1E Kingdom card set tends to focus around the use of Smithy and supporting cards. In addition the varying skill level of human players and relatively small number of player logs may also be impacting the FG1E clustering. It would be interesting to observe how clustering changes as game mechanics change between Kingdom card sets, and as player skill level changes.

D. Application to Other DBCGs and Tabletop Games

It would also be useful to see how these methods translate and what results are seen in other DBCGs. For example, the game *Ascension* is another archetypal DBCG, however it differs from *Dominion* in the existence of factions. These effectively act like suits in the game, with the total card set being split across four factions (plus some nonfactions cards). Cards within factions are expected to have greater levels of synergies, which we would expect to see reflected in playtraces via strategies and clusters that focus on particular factions and combinations of factions.

Taking this one step further, the use of N-Grams-based playtraces generated from player actions is applicable to any game where each player takes a discrete set of actions. Therefore clustering of this type of playtrace could potentially be used to explore the strategy space of a much broader class of tabletop games than DBCGs.

VIII. Conclusion

In this article, we introduce two types of playtraces, which, combined with AI simulated or human play, provide insights into the strategy space of DBCGs. It was shown that using MCTS to simulate game sessions of *Dominion*, we can generate playtrace clusters that highlight different strategies and card synergies.

Playtraces can also help highlight weaker cards. In particular, infrequent card usage may indicate a redundant mechanic. An example is the Workshop card, which along with the Woodcutter saw very little use during both (unrestricted) MCTS and human play, see Fig. 9. Playtraces also reveal synergies, such as the Gardens/Workshop combination and tactics like using Smithy to draw treasures for Gold and Province purchases.

Distinct clustering of playtraces into strategy groupings was observed, particularly in the SD Kingdom card set case. In both the card count and N-Gram playtrace cases, the BMWG and DW strategies from FSM agents were grouped into separate clusters. For MCTS, clusters highlighted Gardens/Workshop and Big Money with Gardens strategies. For FG1E, clustering was not observed and play tended to focus around a dominant strategy of building a Smithy-based engine.

Finally, when drawing conclusions from our playtraces, we must consider the strengths and weakness of our AI agent, and use of restricted play can help in exploring the space of strategies available in a game.

While this study focuses on a single DBCG, the methods, especially the game-agnostic N-Gram approach, could support strategic analysis in other DBCGs and tabletop games.

Acknowledgment

This paper is an abridged version of a master's dissertation. The author would like to thank his supervisor, Raghubir Singh, for valuable comments, as well as Simon Lucas for suggesting N-Grams, James Goodman for his TAG implementation of Dominion and general feedback, and Raluca Gaina and Diego Perez-Liebana for their support with TAG. The author would also like to thank Doug Zongker for providing archived human player logs from Dominion.isotropic.org.

References

[1] B. Ingram, B. Rosman, C. v. Alten, and R. Klein, "Play-style identification through deep unsupervised clustering of trajectories," in *Proc. 2022 IEEE Conf. Games*, 2022, pp. 393–400.

[2] Y.-E Liu, E. Andersen, R. Snider, S. Cooper, and Z. Popovic, "Feature-based projections for effective playtrace analysis," in *Proc. Int. Conf. Foundations Digit Games*, 2011, Art no. 14029159. [Online]. Available: https://api.semanticscholar.org/CorpusID

[3] J. Campbell, J. Tremblay, and C. Verbrugge, "Clustering player paths," in *Proc. Int. Conf. Foundations Digit. Games*, 2015, Art no. 18136549. [Online]. Available: https://api.semanticscholar.org/CorpusID

[4] J. C. Osborn, B. Samuel, and M. Mateas, "Visualizing the strategic landscape of arbitrary games," *Inf. Visual.*, vol. 17, no. 3, pp. 196–217, 2018, doi: 10.1177/1473871617718377.

[5] H. Bendekgey, "Clustering player strategies from variable-length game logs in dominion," 2018, *arXiv:1811.11273*.

[6] T. Mahlmann, J. Togelius, and G. Yannakakis, "Evolving card sets towards balancing dominion," in *Proc. IEEE Congr. Evol. Comput.*, 2012, pp. 1–8.

[7] C. Ford and M. Ohata, "Game balancing in dominion: An approach to identifying problematic game elements," in *Proc. AAAI Conf. Artif. Intell.*, Jun. 2022, pp. 12744–12750. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/21552

[8] N. Budijono et al., "LUDUS: An optimization framework to balance auto battler cards," in *Proc. AAAI Conf. Artif. Intell.*, Jun. 2022, pp. 12727–12734. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/21550

[9] C. Aggarwal, A. Hinneburg, and D. Keim, "On the surprising behavior of distance metric in high-dimensional space," in *Proc. 8th Int. Con. Database Theory*, London, U.K., 2002, pp. 420–434.

[10] D. Hiemstra, *N-Gram Models*. Boston, MA, USA: Springer, 2009, pp. 1910–1910, doi: 10.1007/978-0-387-39940-9_935.

[11] J. MacQueen et al., "Some methods for classification and analysis of multivariate observations," in *Proc. 5th Berkeley Symp. Math. Statist. Probability*, Oakland, CA, USA, 1967, pp. 281–297.

[12] L. Kaufman and P. J. Rousseeuw, *Partitioning Around Medoids (Program PAM)*. Hoboken, NJ, USA: Wiley, 2008, pp. 68–125, doi: 10.1002/9780470316801.ch2.
[13] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. Int. Conf. Knowl. Discov. Data Mining*, 1996, pp. 226–231. [Online]. Available: https://api.semanticscholar.org/CorpusID
[14] A. Mead, "Review of the development of multidimensional scaling methods," *J. Roy. Stat. Soc.*, vol. 41, no. 1, pp. 27–39, 1992. [Online]. Available: http://www.jstor.org/stable/2348634
[15] P. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *J. Comput. Appl. Math.*, vol. 20, no. 1, pp. 53–65, Nov. 1987. [Online]. Available: http://svn.donarmstrong.com/don/trunk/projects/research/papers_to_read/statistics/silhouettes_a_graphical_aid_to_the_interpretation_and_validation_of_cluster_analysis_rousseeuw_j_comp_app_math_20_53_1987.pdf
[16] R. D. Gaina, M. Balla, A. Dockhorn, R. Montoliu, and D. Perez-Liebana, "TAG: A tabletop games framework," in *AIIDE Workshops* 2020. Available: https://api.semanticscholar.org/CorpusID:229171680
[17] C. B. Browne et al., "A survey of Monte Carlo tree search methods," *IEEE Trans. Comput. Intell. AI Games*, vol. 4, no. 1, pp. 1–43, Mar. 2012.
[18] M. Balla, G. E. M. Long, D. Jeurissen, J. Goodman, R. D. Gaina, and D. Perez-Liebana, "PyTAG: Challenges and opportunities for reinforcement learning in tabletop games," IEEE Conference on Games (CoG), 2023, pp. 1-8. Available: https://api.semanticscholar.org/CorpusID:259982939
[19] J. Goodman, D. Perez-Liebana, and S. Lucas, "Measuring randomness in tabletop games," in *Proc. 2024 IEEE Conf. Games*, 2024, pp. 1–8.
[20] J. Goodman, D. Perez Liebana, and S. Lucas, "Following the leader in multiplayer tabletop games," in *Proc. 18th Int. Conf. Foundations Digital Games*, 2023, pp. 1–11.
[21] M. D. Fisher, "Provincial: A kingdom-adaptive AI for dominion," Stanford Univ., Stanford, CA, USA, 2014. Available: https://graphics.stanford.edu/∼mdfisher/DominionAI.html
[22] P. I. Cowling, C. D. Ward, and E. J. Powley, "Ensemble determinization in Monte Carlo tree search for the imperfect information card game magic: The gathering," *IEEE Trans. Comput. Intell. AI Games*, vol. 4, no. 4, pp. 241–257, Dec. 2012.
[23] J. Goodman, D. Perez-Liebana, and S. Lucas, "Multitree mcts in tabletop games," in *Proc. 2022 IEEE Conf. Games*, 2022, pp. 292–299.
[24] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Mach. Learn.*, vol. 47, no. 2–3, pp. 235–256, May 2002, doi: 10.1023/A:1013689704352.
[25] H. Finnsson and Y. Björnsson, "Simulation-based approach to general game playing," in *Proc. AAAI Conf. Artif. Intell.*, 2008, pp. 259–264.
[26] A. Jaffe, "Understanding game balance with quantitative methods," Ph.D. thesis, Computer Science and Engineering, Univ. Washington, Seattle, WA, USA, Jul. 2013 [Online]. Available: https://digital.lib.washington.edu/researchworks/handle/1773/22797
[27] H. Yang and Y. Kuo, "An AI for dominion using deep reinforcement learning," Accessed: Jul. 12, 2023. [Online]. Available: http://cs230.stanford.edu/projects_fall_2019/reports/26260348.pdf