International Journal of Computational Intelligence Systems
Vol. 13(1), 2020, *pp.* 1207–1217
DOI: https://doi.org/10.2991/ijcis.d.200805.001; ISSN: 1875-6891; eISSN: 1875-6883
https://www.atlantis-press.com/journals/ijcis/

Research Article

Predicting Cards Using a Fuzzy Multiset Clustering of Decks

Alexander Dockhorn[1,*], Rudolf Kruse[2]

[1] *School of Electronic Engineering and Computer Science, Queen Mary University of London, Mile End Road, London, E1 4NS, United Kingdom*
[2] *Faculty of Computer Science, Otto von Guericke University Magdeburg Universitätsplatz 2, Magdeburg, 39106, Germany*

ARTICLE INFO

Article History
Received 29 Apr 2020
Accepted 31 Jul 2020

Keywords
Fuzzy multisets
Clustering
Deck analysis
Hearthstone

ABSTRACT

Search-based agents have shown to perform well in many game-based applications. In the context of partially-observable scenarios agent's require the state to be fully determinized. Especially in case of collectible cards games, the sheer number of decks constructed by players hinder an agent to reliably estimate the game's current state, and therefore, renders the search ineffective. In this paper, we propose the use of a (fuzzy) multiset representation to describe frequently played decks. Extracted deck prototypes have shown to match human expert labels well and seem to serve as an efficient abstraction of the deck space. We further show that such deck prototypes allow the agent to predict upcoming cards with high accuracy, therefore, allowing more accurate sampling procedures for search-based agents.

1. INTRODUCTION

Automated game playing poses many interesting challenges to the development of artificial intelligence (AI) agents. While many studies presented good results on full information games, the agent's performance is often restricted by partial information on the current game state. The online game Hearthstone: Heroes of Warcraft (in short Hearthstone) [1] is such a partial information game, which is currently very popular among players [2]. Hearthstone is a deck-building card game in which players create their own decks to play against each other. At the time of writing this paper, Hearthstone includes roughly 2500 cards of which players choose a deck of 30 cards to face their opponent. While this paper will only discuss rules and characteristics of Hearthstone that influence the deck building, the interested reader is referred to some excellent resources on the web [1,3], which offer comprehensive reviews of the game's mechanics.

During a game, players do not know their opponent's deck nor hand cards, but observe newly played cards every turn. Since the game state is partially unknown, missing information needs to be completed by the agent before it can apply any search-based algorithm to determine its actions. Similar to other games, expert players of Hearthstone are able to predict opponent moves to a certain degree. This ability can be attributed to their knowledge of the current meta-game, which can be understood as the likelihood of decks and strategies an opponent may play. When building a deck, users often combine cards and their effects with a certain strategy in mind. Players of the game often refer to the concept of deck archetypes, when they discuss decks with a common theme and similar card sets. Such a deck archetype may develop due to the popularity of a certain strategy and its accompanied deck. However, many players do not own all the necessary cards of a specific deck they are trying to build, therefore, variants of these deck archetypes can exist.

In the context of research, the annual Hearthstone AI competition compares agents based on their ability to play the game in various game modes [4]. It has motivated many interesting works, which often focus on the agent's ability to optimize its own turn. Nevertheless, winning approaches also attempt to model the opponent's general strategy [5,6].

In regard of the creation of an autonomous agent, estimating the opponent's upcoming actions is crucial in choosing their own actions. However, due to the opponent's cards being hidden, agents are limited in their capability of predicting these moves. Algorithms like Information Set Monte Carlo Tree Search (MCTS) [7,8] try to handle this problem by randomly sampling the opponent's hand cards to create a determinization of the game state. Based on the generated determinization the search process simulates its own and its opponent's actions and chooses the most promising action. Nevertheless, due to the large number of available cards, the accuracy of this sampling strategy is fairly low and can therefore limit the agent's success. To reduce the sampling size, it has been suggested to use a data base of frequently played decks. Agents using this sampling strategy have resulted in agents with very high performance [9,10] when being compared to other game-playing agents. Nevertheless, they require frequent updates of their deck data base in case the decks' popularity changes over time.

Sievers and Helmert [11] developed an extended version of Information Set MCTS, which does not sample a single determinization, but multiple determinizations of the game state. For each of these determinizations, the algorithm performs a separate run of MCTS.

[*] *Corresponding author. Email: a.dockhorn@qmul.ac.uk*

The results of each run are later aggregated to determine the agent's next action. Sievers and Helmert evaluated their approach on the game Doppelkopf, in which player's need to guess their teammates during the first turns of every game. This critical guess can have a high impact on the following actions. Their approach showed to be valuable in estimating the risk of the next action and improved the agent's overall performance. A study by Dockhorn *et al.* [12] further extended this approach by using neural networks for guiding the simulation, therefore, improving the quality of simulations during the search process and its result.

In a recent paper, we introduced an autonomous agent for Hearthstone [6], which implements a similar approach to the one presented in Reference [12]. We observed that the prediction accuracy and the agent's game-playing performance is still limited, which seems to be due to the enormous number of possible game state determinizations. We reduced the game state's sample space by using information of previously played cards to predict likely cards on the opponent's hand [13]. Using card co-occurrences of previously seen cards and the opponent's hand cards we were able to sample game state determinizations with a higher likelihood. We observed an improvement of the agent's game-playing performance in comparison to a uniform sampling. However, agents that were given complete information still outperformed the proposed agent. From this, we infer that further increasing the accuracy of the game state sampling may in turn further improve the agent's performance.

In the pursuit of creating a better agent for Hearthstone, we want to enhance the agent's prediction capabilities by modeling deck archetypes. In Section 2 we review restrictions of the deck-building process and provide a short overview of the theory of (fuzzy) multisets and various clustering approaches. In the subsequent Section 3, these methods will be used to develop a theoretical model of deck archetypes and how to mine them from a database of recent games. Especially the advantages of using fuzzy multisets instead of crisp multisets are highlighted based on some explanatory examples. We further present a case study, which is based on extracting deck archetypes and their centroid representation from actual playing data of the game Hearthstone (Section 4). We compare our result with a hand-labeled data set and show that the developed approach is able to identify deck archetypes of similar quality.

In this extended version of our conference paper [14] we added a method for predicting upcoming cards based on identified deck archetypes. The proposed sampling approach will be described in Section 5. Finally, we will evaluate the agent's ability to predict upcoming cards in the opponent's deck (Section 6). The paper concludes with a short analysis of the proposed approach and its possible application to the development of game-playing agents.

2. PRELIMINARIES

We begin this section with a short overview of deck archetypes and how they are defined in Hearthstone. A complete description of the game Hearthstone will be beyond the scope of this paper. For this reason, we would like to refer the interested reader to some excellent web sources maintained by developer [1] and the community [3]. We further provide detailed explanations on (fuzzy) multisets and selected clustering algorithms, which will be used to model and mine deck archetypes in Section 3.

2.1. Deck Building and Deck Archetypes

In Hearthstone a deck is a set of 30 cards, which can be chosen out of the roughly 2500 cards currently available in the game. Each card offers certain effects, which can be used to affect the current game state. Some of these effects create useful synergies that players try to exploit during the game, e.g., attack with a minion card, which will get damaged during the fight, and follow up by healing this minion using a spell card. The choice of cards to be put in the deck is restricted by a small set of rules:

- players can only include cards they currently own (which are unlocked on their player's account)
- a deck belongs to one out of 9 hero classes who are limited to a subset of about 400 cards each
- a deck can only include cards that are either neutral or specific to the chosen hero class
- depending on its rarity, a card can be included either once or twice
- a deck can be made for a specific game mode that adds additional restrictions, e.g., standard or arena.

While players can create a large number of different decks not all of them are equally successful. The most successful decks define the meta and get known as meta-decks. Often these meta-decks will spawn multiple variants in which players replace just a few cards without changing the main theme of the deck.

A deck archetype describes the resulting cluster of decks with common card subsets. In this work, we will distinguish included cards into two groups, namely core and variant cards. While the core of a deck archetype contains cards that are included in all instances of this archetype, the inclusion of variant cards depends on the given instance. Core cards often define the main building blocks of the archetype and its accompanying strategy. In contrast, variant cards are often included to compensate for cards that the player does not own or to reflect the player's personal preferences. The following subsection will introduce (fuzzy) multisets, which will later be used to model deck archetypes.

2.2. (Fuzzy) Multisets/Bags

Multisets (also called bags) are collections of objects in which an object can be represented multiple times. In this paper, we will closely follow the notation introduced by Miyamoto [15] and Yager [16]. We denote a multiset by

$$M = \{C_M(x)/x \;:\; x \in X\} \tag{1}$$

in which X is the set of elements that can be included and C_M a function that maps each object x_i to its number of copies n_i in M:

$$C_M : X \to \mathbb{N} \qquad C_M(x_i) = n_i \tag{2}$$

The collection of all possible multisets of a universal set X is denoted by $\mathcal{M}(X)$.

For comparing two multisets L and M inclusion is defined by

$$L \subseteq M \text{ iff } C_L(x) \leqslant C_M(x) \text{ holds } \forall x \in X \tag{3}$$

and (as a consequence) equality is given by

$$L = M \text{ iff } C_L(x) = C_M(x) \text{ holds } \forall x \in X \tag{4}$$

Union, intersection, and addition are defined pointwise for all $x \in X$ by

$$C_{L\cup M}(x) = C_L(x) \vee C_M(x) \tag{5}$$

$$C_{L\cap M}(x) = C_L(x) \wedge C_M(x) \tag{6}$$

$$C_{L\oplus M}(x) = C_L(x) + C_M(x) \tag{7}$$

where $\vee$ and $\wedge$ imply the max and min operators.

A fuzzy extension of multisets was first introduced by Yager (using the term fuzzy bags) [16]. Here, the sample fuzzy multiset

$$A = \{(x, 0.5),\ (x, 0.3),\ (y, 1),\ (y, 0.5),\ (y, 0.2)\}$$

denotes the occurrence of each object and its membership degree.

For simplicity we group objects of the same kind and their membership degrees, such as in

$$A = \{(0.5, 0.3)/x,\ (1, 0.5, 0.2)/y\}$$

in which the memberships $\{0.5, 0.3\}$ and $\{1, 0.5, 0.2\}$ correspond to the objects x and y, respectively. Therefore, in fuzzy multisets $C_A(x)$ is a finite multiset of the unit interval [16].

For each object x we further define the membership sequence to be the decreasingly-ordered sequence of elements in $C_A(x)$. We will make use of the standard form introduced by Miyamoto [15]:

$$(\mu_A^1(x), \dots, \mu_A^p(x)), \qquad \mu_A^1(x) \geqslant \dots \geqslant \mu_A^p(x) \tag{8}$$

Let $L(x; A)$ be the length of the membership sequence $(\mu_A^1(x), \dots, \mu_A^p(x))$ of multiset A be denoted by

$$L(x; A) = \begin{cases} \max\{j \ :\ \mu_A^j(x) \neq 0\}, & \text{if } x \in A \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

Any operation between two multisets A and B requires the membership sequences of each object to be of equal length. We define the length $L(x; A, B)$ of the resulting membership sequence to be

$$L(x; A, B) = \max\{L(x; A),\ L(x; B)\} \tag{10}$$

For the sake of simplicity we assume a membership degree of

$$\mu_A^i(x) = 0; \quad \forall i \text{ with } L(x; A) < i \leqslant L(x; A, B) \tag{11}$$

in case the object x is included less than $L(x; A, B)$ times in the multiset A (likewise for B).

Similar to crisp multisets we can define inclusion, equality, union, and intersection based on the membership sequences of each element. Let A and B be two fuzzy multisets.

$$A \subseteq B \text{ iff } \mu_A^j(x) \leqslant \mu_B^j(x) \text{ holds for } j = 1, 2, \dots, L(x; A, B),\ \forall x \in X \tag{12}$$

$$A = B \text{ iff } \mu_A^j(x) = \mu_B^j(x) \text{ holds for } j = 1, 2, \dots, L(x; A, B),\ \forall x \in X \tag{13}$$

Similarly, union and intersection are defined pointwise for all $x \in X$ by

$$\mu_{A\cup B}^j = \mu_A^j(x) \vee \mu_B^j(x) \quad j = 1, 2, \dots, L(x; A, B) \tag{14}$$

$$\mu_{A\cap B}^j = \mu_A^j(x) \wedge \mu_B^j(x) \quad j = 1, 2, \dots, L(x; A, B) \tag{15}$$

To clarify the notation we provide the following short example. Consider the two fuzzy multisets A and B over the set of objects $\{x, y, z\}$:

$$\begin{aligned} A &= \{(0.5,\ 0.2)/x,\ (1.0,\ 0.5,\ 0.2)/y\} \\ B &= \{(1.0)/x,\ (0.7,\ 0.6)/y,\ (0.9, 0.5)/z\} \end{aligned}$$

The length per object is

$$L(x; A, B) = 2; \quad L(y; A, B) = 3; \quad L(z; A, B) = 2$$

For simplicity we extend the membership sequences for both multisets according to the maximal observed length:

$$\begin{aligned} A &= \{(0.5,\ 0.2)/x,\ (1.0,\ 0.5,\ 0.2)/y,\ (0.0, 0.0)/z\} \\ B &= \{(1.0,\ 0.0)/x,\ (0.7,\ 0.6,\ 0.0)/y,\ (0.9, 0.5)/z\} \end{aligned}$$

Based on the extended membership sequences we can determine union and intersection of both multisets:

$$A \cup B = \{(1.0,\ 0.2)/x,\ (1.0,\ 0.6,\ 0.2)/y,\ (0.9, 0.5)/z\}$$

$$A \cap B = \{(0.5)/x,\ (0.7,\ 0.5)/y\}$$

2.3. Clustering Algorithms

In this work, we are going to present the results of our deck archetype clustering process. The interested reader is referred to specialized literature on the topics of data mining and (fuzzy) cluster analysis, which provide a more comprehensive review of alternative approaches than this paper could offer [17–21].

2.3.1. Partitional clustering

The clustering algorithm k-means [22] is the most common representative of partitional clustering algorithms. During initialization, k cluster prototypes are randomly generated or selected from the set of available data points. The cluster prototypes are iteratively updated to better represent and partition the points in the data set. For this purpose, each data point is assigned to its closest prototype. In a second step, the prototypes are moved to the center of all assigned points to minimize the sum of squared errors between a prototype and all its assigned data points. Given a clustering $\mathcal{C} = \{\mathbb{C}_1, \dots, \mathbb{C}_K\}$, let the sum of squared error be calculated by

$$SSE(\mathcal{C}) = \sum_{k=1}^{K} \sum_{x_i \in \mathbb{C}_k} |x_i - \langle \mathbb{C}_k \rangle|^2, \tag{16}$$

such that $\langle \mathbb{C}_k \rangle$ represents the centroid of cluster $\mathbb{C}_k$. Due to its scoring function, k-means favors clusters that are compact and well separated.

2.3.2. Hierarchical agglomerative clustering

Hierarchical agglomerative clustering is a class of bottom-up clustering algorithms, in which each data point is assigned to a unique cluster during initialization. These clusters are iteratively merged according to a linkage criterion. The merge process is repeated until a minimum number of clusters is reached or all data points belong to a common cluster.

In this work we will make use of the following linkage criteria, which both determine the distance of two clusters based on the distances of points contained in differing clusters:

- **single linkage** [23] reports the minimal distance between two points of different clusters

$$d_{\text{single}}(\mathbb{C}_i, \mathbb{C}_j) = \min_{a \in \mathbb{C}_i,\ b \in \mathbb{C}_j} d(a, b) \tag{17}$$

- **complete linkage** [24] reports the maximal distance between two points of different clusters

$$d_{\text{complete}}(\mathbb{C}_i, \mathbb{C}_j) = \max_{a \in \mathbb{C}_i,\ b \in \mathbb{C}_j} d(a, b) \tag{18}$$

Similar to k-means, the complete linkage favors compact and well-separated clusters. In contrast, the single linkage criterion may exhibit a phenomenoncalled chaining in which existing clusters are merged due to having two points, which are close to each other, even though other points are far apart.

2.3.3. Density-based clustering

The Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm proposed by Ester *et al.* [25], forms clusters by searching for dense regions of points. The ε-neighborhood of a point consists of all points with a maximal distance of ε:

$$N_\varepsilon(p) = \{q \in X \mid d(p, q) \leqslant \varepsilon\} \tag{19}$$

The region around a point is considered to be dense in case the number of points in its ε—neighborhood exceeds the threshold m_{Pts}. Points that satisfy this condition are called core points. A point q is directly density-reachable from point p, if $q \in N_\varepsilon(p)$ and p is a core-point. The transitive closure of directly density-reachable points is called density-reachable. Finally, two points are density-connected when a point exists from which both are density-reachable. A cluster is described by the maximal set of points that are density-connected to each other.

3. (FUZZY) MULTISET ANALYSIS OF DECK ARCHETYPES

In the previous section, we discussed various building components for the deck archetype mining algorithm we are proposing in this section. We will first define how a deck archetype can be represented in terms of fuzzy multisets and further describe a mining routine based on the clustering algorithms presented above.

A natural representation of a deck is a multiset of cards.

$$D = \{C_D(x)/x \mid x \in X\} \tag{20}$$

where $C_D(x) \in \mathbb{N}$ is the number of inclusions of card x in deck D. Due to the restrictions of Hearthstone's deck-building process, any card can be included twice or less $C_D(x) \leqslant 2$. It is also known that each deck has exactly 30 cards, which is equal to the sum of object counts or membership degrees in the deck.

3.1. Modelling Deck Archetypes

The Hearthstone community defines a deck archetype to be a collection of decks with a common set of cards. In the following, we are going to model a deck archetype to be the representative of a cluster of decks.

Let's consider two crisp decks D_1 and D_2 over the set of elements $X = \{a, b, c, d, e, f\}$ of the form

$$\begin{aligned} D_1 &= \{1/a,\ 1/b,\ 2/c,\ 1/d,\ 0/e,\ 2/f\} \\ D_2 &= \{1/a,\ 1/b,\ 2/c,\ 0/d,\ 2/e,\ 1/f\} \end{aligned}$$

The intersection $M_{D_1 \cap D_2}$ of these two decks is the multiset:

$$M_{D_1 \cap D_2} = \{1/a,\ 1/b,\ 2/c,\ 0/d,\ 0/e,\ 1/f\}$$

While the resulting set describes the core of these two decks, the information of possible variants is lost during the generation of the common subset. A similar problem occurs if we generate the union $M_{D_1 \cup D_2}$ of both decks:

$$M_{D_1 \cup D_2} = \{1/a,\ 1/b,\ 2/c,\ 1/d,\ 2/e,\ 2/f\}$$

While the union operator preserves information on the inclusion of d and e we misleadingly represent these variants, i.e., based on its count in $M_{D_1 \cup D_2}$ variant object d is indistinguishable from the core objects a and b (similar observations can be made for the objects c and e). Hence, objects with different inclusion patterns in D_1 and D_2 are equally represented in the merged multiset. Replacing the union with the addition operation would yield similar problems and also increase the cardinality of the resulting multiset.

For the crisp multiset we define the average multiset $M_{\langle L,M \rangle}$ of two multisets L and M to include the average number of occurrences per object in these multisets and denote it by

$$C_{\langle L,M \rangle}(x) = \frac{C_L(x) + C_M(x)}{2}, \quad \forall x \in X \tag{21}$$

Hence, the average of clusters D_1 and D_2 is

$$M_{\langle D_1, D_2 \rangle} = \{1/a,\ 1/b,\ 2/c,\ 0.5/d,\ 1/e,\ 1.5/f\}$$

While the average operator already clearly distinguishes the inclusion patterns for a, b, and d, we can still observe problems with varying numbers of inclusion, e.g., a and e. However, extending the representation to fuzzy multisets can help to solve this problem.

For this purpose, we transfer the average operator for crisp multisets to fuzzy multisets by calculating the average of every element of an object's grade sequence. Thus, the average operator for two fuzzy multisets A and B can be denoted by

$$\mu^i_{\langle A,B \rangle}(x) = \frac{\mu^i_A(x) + \mu^i_B(x)}{2}, \quad i = 1, \ldots, p,\ \forall x \in X \tag{22}$$

Representing both decks as fuzzy multisets results in the following centroid:

$$\begin{aligned} D_1 &= \{(1)/a,\ (1)/b,\ (1,1)/c,\ (1)/d,\ (0)/e,\ (1,1)/f\} \\ D_2 &= \{(1)/a,\ (1)/b,\ (1,1)/c,\ (0)/d,\ (1,1)/e,\ (1)/f\} \end{aligned}$$

$$\begin{aligned} M_{\langle D_1, D_2\rangle} = \{&(1)/a,\ (1)/b,\ (1,1)/c,\ (0.5)/d, \\ &(0.5,0.5)/e,\ (1.0,0.5)/f\} \end{aligned}$$

To ensure a stable clustering process we want to adjust the definition of the (fuzzy) multiset centroid to fulfill the associative property, since the result of merging multiple multisets should be independent of their merging order, specifically we want the following properties to be fulfilled:

$$\begin{aligned} C_{\langle\langle D_1,D_2\rangle,D_3\rangle}(x) &= C_{\langle D_1,\langle D_2,D_3\rangle\rangle}(x), \quad \forall x \in X \\ M_{\langle\langle D_1,D_2\rangle,D_3\rangle} &= M_{\langle D_1,\langle D_2,D_3\rangle\rangle} \end{aligned} \tag{23}$$

Let a cluster $\mathbb{C}$ be a multiset over the set $\{M_1, \ldots, M_n\}$ of multisets over the set of objects X. The centroid $\langle\mathbb{C}\rangle$ of cluster $\mathbb{C}$, which is itself a multiset over the set of objects X, should be independent of the order of inclusion of said multisets, thus fulfilling the associative property of the order of merges. For this purpose, we generalize the average operator of two multisets (Equation 21) to take the number of inclusions per multiset into account:

$$C_{\langle\mathbb{C}\rangle}(x) = \frac{\sum_{M_i\in\mathbb{C}} C_{M_i}(x)\cdot C_{\mathbb{C}}(M_i)}{\sum_j C_{\mathbb{C}}(M_j)}, \quad \forall x \in X \tag{24}$$

The same can be done for a cluster of fuzzy multisets

$$\begin{gathered} \mu^k_{\langle\mathbb{C}\rangle}(x) = \frac{\sum_{M_i\in\mathbb{C}} \mu^k_{M_i}(x)\cdot C_{\mathbb{C}}(M_i)}{\sum_j C_{\mathbb{C}}(M_j)}, \\ k = 1, \ldots, p, \quad \forall x \in X \end{gathered} \tag{25}$$

We will use the cluster centroid to represent the cluster and all its contained decks in a single (fuzzy) multiset.

3.2. Clustering of (Fuzzy) Multisets

For mining deck archetypes we are going to apply the clustering algorithms introduced in Section 2.3 to a data set of recently played decks. Since these clustering methods are distance-based we need a suitable distance measure to group data points into clusters of similar objects.

To measure the distance of two multisets L and M we define their Euclidean distance by

$$d_{\text{euclid}}(L, M) = \left(\sum_{x\in X} (C_L(x) - C_M(x))^2\right)^{\frac{1}{2}} \tag{26}$$

and transfer the definition to be applied to fuzzy multisets A and B:

$$d_{\text{euclid}}(A, B) = \left(\sum_{x\in X} \sum_{i=1}^{L(x;A,B)} \left(\mu^i_A(x) - \mu^i_B(x)\right)^2\right)^{\frac{1}{2}} \tag{27}$$

In our work, we will compare results based on the Euclidean distance with results obtained from applying the Jaccard distance measure. Here we use the general Jaccard distance [26]:

$$d_{\text{jaccard}}(X, Y) = 1 - \frac{|X \cap Y|}{|X \cup Y|} = 1 - \frac{\sum_i \min(x_i, y_i)}{\sum_i \max(x_i, y_i)} \tag{28}$$

and apply it to two multisets L and M:

$$d_{\text{jaccard}}(L, M) = 1 - \frac{\sum_{x\in X} \min(C_L(x), C_M(x))}{\sum_{x\in X} \max(C_L(x), C_M(x))} \tag{29}$$

Similar to the Euclidean distance we transfer the equation to measure the distance of two fuzzy multisets A and B:

$$d_{\text{jaccard}}(A, B) = 1 - \frac{\sum_{x\in X} \sum_{j=1}^{L(x;A,B)} \mu^j_{A\cap B}}{\sum_{x\in X} \sum_{j=1}^{L(x;A,B)} \mu^j_{A\cup B}} \tag{30}$$

4. CLUSTERING EVALUATION

All project files for the following evaluation are available at Github (see Ref. [27]). We evaluated our clustering approach using deck data of the HSReplay website [28]. The website offers easy access to a large collection of recently played games. Players can choose to use the Hearthstone Deck Tracker plugin, which automatically records played games and uploads them to the HSReplay servers. In return, players can access information on the probability of their opponent's cards while playing the game.

We have extracted a deck data set that contains data from February 5th to 20th 2019. Each deck entry stores the deck's cards, the total amount of games recorded during the two weeks, its average win-rate, average game-length, and average turn count. Additionally, each deck entry provides information on the suggested deck archetype, which was labeled by expert players. The deck data set consists of a total of 956 distributed over 9 hero classes (cf. Table 1).

We will compare the result of each clustering method with the labeling provided in the data set. For this purpose, we use the external validation measures homogeneity, completeness, and v-measure [29]. While homogeneity is satisfied in case the clusters contain only data points that are members of a single class (as labeled in the ground truth), completeness is satisfied if all the data points that are members of a given class are elements of the same cluster. The v-measure is the harmonic mean of homogeneity and completeness. All three measures provide values between 0.0 and 1.0, where larger values are desirable.

We first calculated the distance matrix of decks of the same hero class for each of the nine heroes. Figure 1 shows the heat map plot of the Jaccard distance of all Druid decks (70) contained in the data set encoded as fuzzy multisets. Clusters of similar groups are clearly distinguishable. Euclidean distance looks similar but is not limited to a range of 0.0 to 1.0, which makes it harder to define a cutoff threshold for stopping the clustering process.

For each clustering algorithm we perform a parameter search. For k-means we varied the number of clusters k and repeated the clustering 10 times with varying initializations. The best-performing clustering in terms of SSE will be reported as the final clustering result. In case of the hierarchical clusterings, we first create a

Table 1 Distribution of hero classes in used data sets.

Hero	nr Games in	
Class	Deck Data Set	Replay Data Set
Druid	70	191
Hunter	171	264
Mage	118	344
Paladin	120	187
Priest	183	542
Rogue	89	205
Shaman	18	287
Warlock	108	243
Warrior	79	799
Total	956	3062

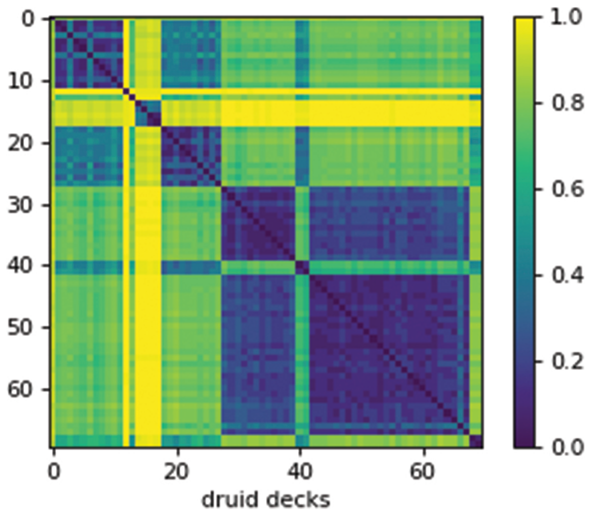


Figure 1 Jaccard distance matrix of Druid decks.

full hierarchy and horizontally cut it at each layer of the hierarchy. Finally, for DBSCAN we use the m_{Pts}-DBSCAN algorithm [30] to quickly generate a hierarchy of clusters for all relevant parameterizations of the ε-radius. We repeat this process for $m_{Pts} = 2$, $m_{Pts} = 5$, and $m_{Pts} = 10$. For each retrieved clustering we report homogeneity, completeness, and the v-measure in Figure 2. Additionally, the best-performing parameterization per clustering in terms of the v-measure is reported in Table 2. Presented results are calculated using the Jaccard distance. Using the Euclidean distance yielded similar results and will therefore be omitted.

The results show that the best clustering result in terms of the v-measure was achieved by single linkage (0.949) closely followed by complete linkage (0.945). Both hierarchical methods, achieve result in relatively stable v-measures for in the range of $[70 - 130]$ clusters. Since Hearthstone's meta-game is dynamic, the number of played deck archetypes will be varying over time. Their stable clustering performance makes these two methods relatively robust against these changes in the meta-game. In comparison, k-means achieves a peak performance for 50 clusters. Finally, DBSCAN with $m_{Pts} = 2$ performs similar, but is very sensitive to the chosen ε value and its resulting number clusters. This problem gets worse for higher m_{Pts} values and is likely to be a result of the high dimnensionality of the input space.

5. PREDICTING UPCOMING CARDS

In the previous sections we have shown how a clustering of decks that resembles a human labeling can be extracted from a data set of played games. In the following, we will propose a method to predict upcoming cards based on a given clustering. Knowing the cluster centroids, a prediction of the opponent's cards can be made using the multi-process described in the following subsections.

5.1. Keeping Track of Observed Cards

At the beginning of the game the agent starts with an empty fuzzy multiset. During the opponent's turn, the agent keeps track of all the opponent's actions. Each card played is added to the agent's fuzzy multiset with a membership grade of 1.0. In case the card has previously been played the membership sequence of this card is extended by another entry of 1.0.

5.2. Determine the Most-Likely Deck Archetype

During the agent's turn, the agent first needs to determine the most-likely deck archetype. This can be done by calculating the pair-wise distance between the fuzzy multiset of observed cards and all the cluster centroids, which represent the deck archetypes of the current meta-game. The closest centroid is assumed to be the most-likely deck archetype and will be considered for the card prediction of upcoming cards. In case the agent can handle multiple state determinizations, a distance-weighted sampling can be applied to select one deck archetype per requested determinization. For this purpose, all distances $d(\langle \mathbb{C}_i \rangle, \text{obs})$ between the observation obs and the fuzzy centroids $\langle \mathbb{C}_i \rangle$ of the i-th cluster are first transformed into a similarity value by

$$\text{sim}(\langle \mathbb{C}_i \rangle) = 1 - \frac{d(\langle \mathbb{C}_i \rangle, \text{obs})}{\max_i(d(\langle \mathbb{C}_i \rangle, \text{obs}))} \tag{31}$$

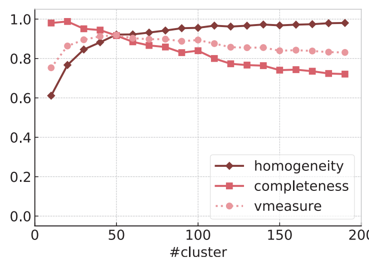
The resulting similarity values are further transformed into a probability distribution by

$$P(\langle \mathbb{C}_i \rangle) = \frac{\text{sim}(\langle \mathbb{C}_i \rangle)}{\sum_{j=1}^{K} \text{sim}\langle \mathbb{C}_j \rangle)} \tag{32}$$

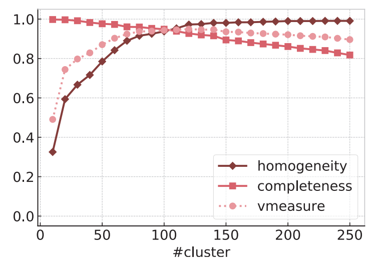
In the upcoming evaluation of the agent's prediction accuracy, we will only consider the most-likely deck archetype during the prediction process. However, in the context of a game-playing agent which uses a search process that can handle an ensemble of state determinizations [11] it may be advantageous to extract cards from a variety of possible archetypes to find a more robust sequence of actions.

5.3. Sample Cards

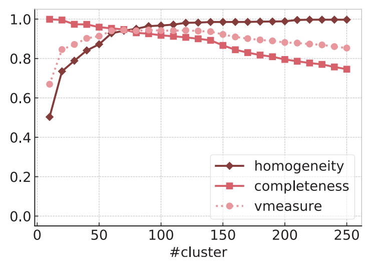
The final step of the prediction process is the sampling of cards. For this purpose, the agent first removes previously observed cards from the centroids membership sequence. For each observed card, the agent removes the highest value from the centroids membership sequence of this card. The remaining entries in the centroid are



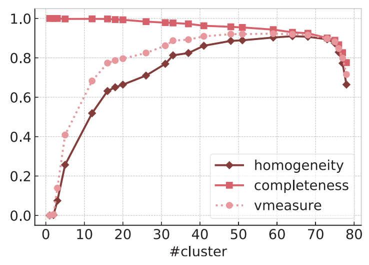
(a) k-means



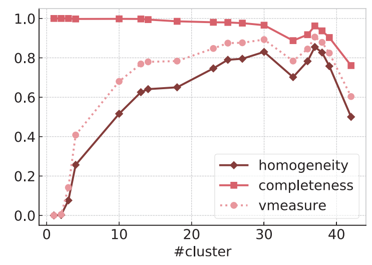
(b) HAC single linkage



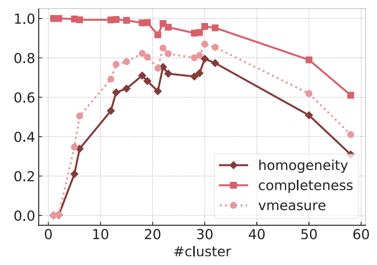
(c) HAC complete linkage



(d) DBSCAN $m_{Pts} = 2$



(e) DBSCAN $m_{Pts} = 5$



(f) DBSCAN $m_{Pts} = 10$

Figure 2 Comparison of clustering results based on external validation measures homogeneity, completeness, and the v-measure.

Table 2 Best-performing parameter configuration per clustering algorithm.

Algorithm	**Parameters**		**Max V-Measure**
k-means	$k = 50$		0.919
HAC single linkage	$k = 120$		0.949
HAC complete linkage	$k = 90$		0.945
m_{Pts}DBSCAN	$m_{Pts} = 2$	$\varepsilon = 0.422$	0.923
m_{Pts}DBSCAN	$m_{Pts} = 5$	$\varepsilon = 0.500$	0.905
m_{Pts}DBSCAN	$m_{Pts} = 10$	$\varepsilon = 0.571$	0.870

ranked according to the sum of their membership sequence. Similar to the bigram-based prediction each card receives a sampling probability based on the determined sum. The resulting probability distribution can be used to sample cards based on their likelihood to appear in the remaining cluster. Sampled cards are temporarily removed from the decks centroid before the next cards are sampled. This process will be repeated until a set of opponent cards has been determined to fully determinize the current game state.

6. EVALUATING THE PREDICTION ACCURACY

Given the prediction process proposed in the previous section, we evaluate the resulting prediction accuracy in two prediction tasks. First, we are testing the agent's ability to predict cards of the remaining game (Section 6.1), and second, we evaluate its accuracy to predict cards of the upcoming turn (Section 6.2). While the former gives us an understanding how well the considered deck archetype matches the opponent's deck, the second scenario is especially relevant when selecting the agent's next actions.

In this evaluation, we will be using a second data set that consists of human player replay data. Each record includes all observed cards that have been played per match but does not contain any information on the remaining cards of both players' decks. Only records that cover games of the standard game mode played during the same patch period (21 February 2019 to 3 April 2019) will be used for evaluating the proposed methods' prediction performance. Used data sets, the source code for the following evaluations, and the raw results can be found in the public git-repository [27].

6.1. Card Prediction for the Remaining Game

In this first evaluation, the accuracy of predicting upcoming cards of the remaining game will be measured. Since the number of observed cards and the complexity of turns changes over time, results will be reported bucketed by turn. Each algorithm is used to predict the 10 most-likely cards after each of the first 10 turns. To assure that the prediction of the last turns can be reasonably tested, only games that lasted at least 15 turns have been selected for this evaluation which resulted in a total of 3062 games. The hero class distribution is shown in (cf. Table 1).

Figure 3 shows the prediction accuracy for the best clustering per algorithm (see Table 2) bucketed by turn. Rank 1 represents the card that is believed to be the most-likely card to be played by the opponent during the remaining game. Vice versa rank 10 is believed to be the 10th most-likely card. Additionally, we report the average accuracy of the 10 cards to be believed the most likely to appear. Since the opponent is not able to play many cards per turn, we also report the aggregated prediction accuracy in Figure 4. This value represents the accuracy of any of the top k-ranked cards being correct.

For all algorithms, the peak performance of predicting the highest-ranked card is achieved in turns 6 or 7. This is likely to be a result of the information gained until this turn, which helps the agent to identify the deck. Since the replay length is limited, we do not know all cards of the opponent's deck. Even in case cards are correctly predicted, chances are that the agent will not be able to observe them due to winning or losing the game before the card can be played.

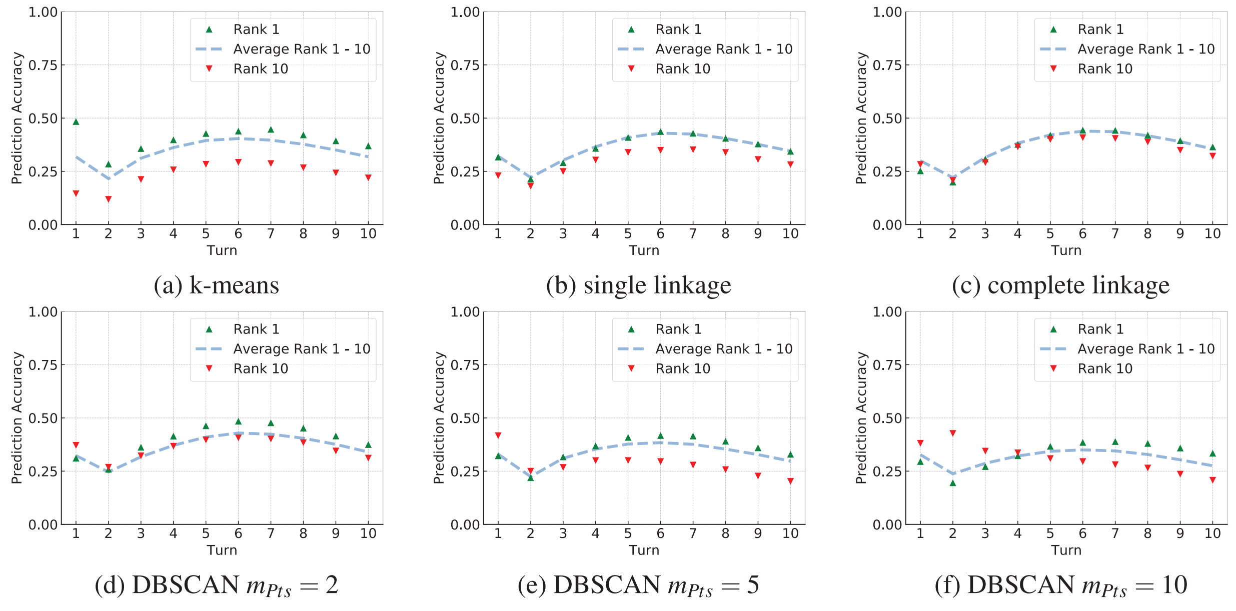


Figure 3 Accuracy for the prediction of cards that may appear in the remaining turns of the game bucketed by turn. Subfigures (a)-(f) show the prediction results based on the clustering result achieved using each algorithm's best-performing parameters. The highest-ranked card is marked in green while the prediction of the 10th ranked is shown in red. The average accuracy of all ten ranks is shown in blue.

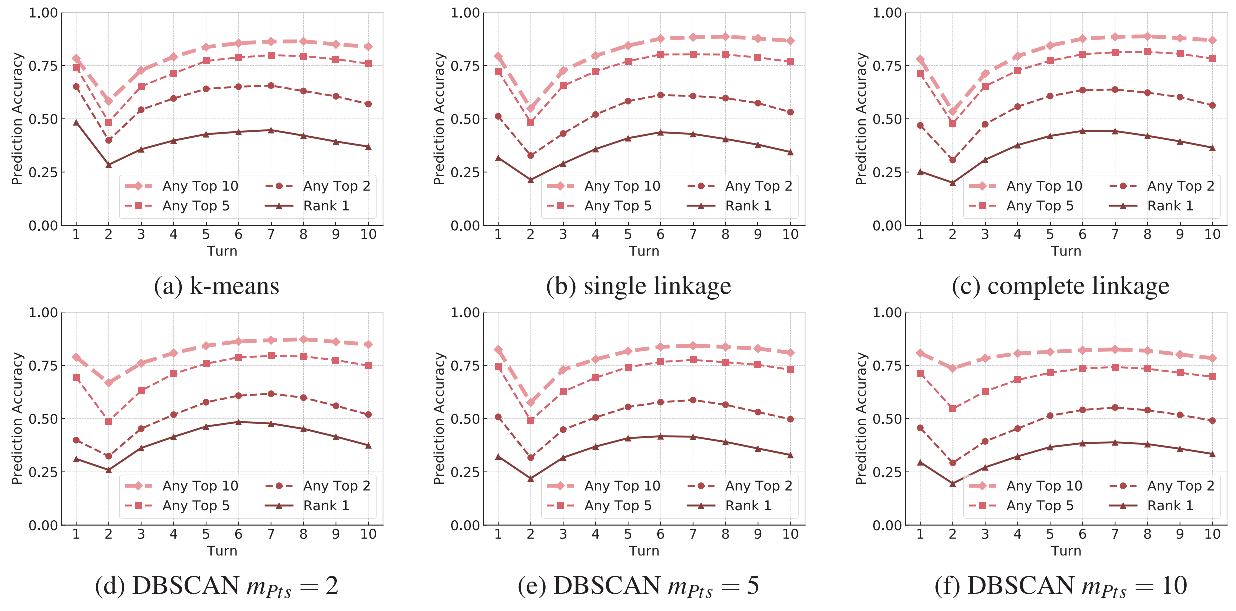


Figure 4 Accuracy for the aggregated prediction of cards that may appear in the remaining turns of the game bucketed by turn. Subfigures (a)-(f) show the prediction results based on the clustering result achieved using each algorithm's best-performing parameters. Reported values describe the accuracy of any of the top ranked cards being correct.

This is likely to be the cause of the steady decline of the prediction accuracy from turn 8 on.

Since multiple cards need to be sampled to create a state determinization of the opponent's hand cards, we also compared the aggregated prediction accuracy of the top k ranks. Values indicate that there is more than a 50% chance that any of the top 2 ranked cards will be seen througout the remainder of the game. Increasing the number of considered cards increases this chance with diminishing returns for higher values of k. Overall, the aggregated top- 10 cards prediction achieves an accuracy of about 80% for all clustering algorithms from the 5th turn onward.

6.2. Card Prediction of the Next Turn

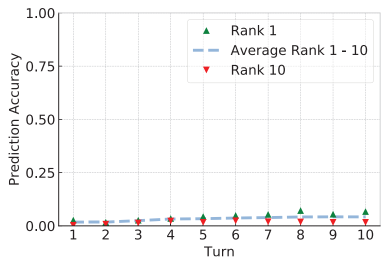
In a similar manner we evaluate the agent's accuracy in predicting cards of the next turn. Naturally, the accuracy will be much lower, since the number of cards that can be played during the next turn is limited.

Figure 5 reports the prediction accuracies of the highest-ranked card, the card with rank 10 and the average of the first 10 ranked cards. While the accuracy during all turns is very low, the aggregated values shown in Figure 6 look more promising. The prediction accuracy of upcoming cards steadily increases until the 9th turn at which it peaks at a value in between 40% and 50% depending on the clustering algorithm used. As a result the agent has a 40%–50% chance to correctly sample a card to be played during the next turn when only considering the most-likely deck.

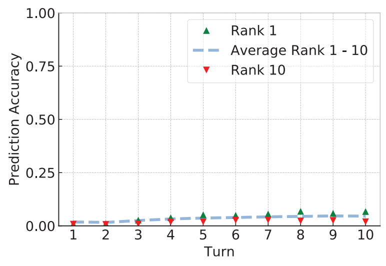
It needs to be noted that the sampling approach does not consider if a card is playable during the next turn. Using an opponent model, the agent still needs to determine which of the sampled cards is likely to be played next.

7. CONCLUSION

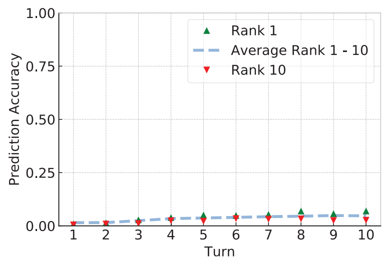
In this paper we reviewed our automatic clustering process for deck archetypes and evaluated it in the context of the collectible card



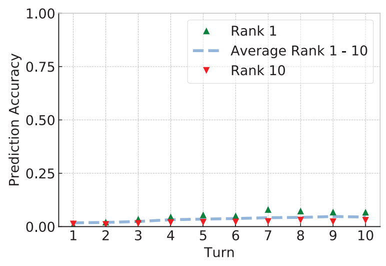
(a) k-means



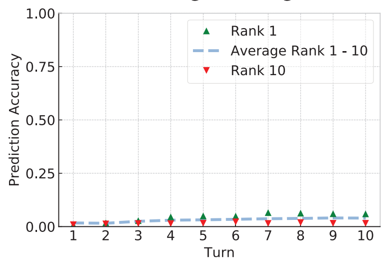
(b) single linkage



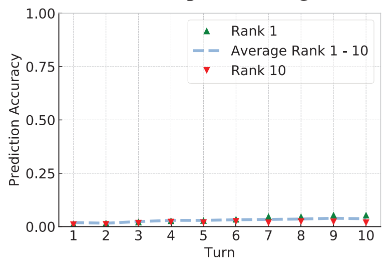
(c) complete linkage



(d) DBSCAN $m_{Pts} = 2$

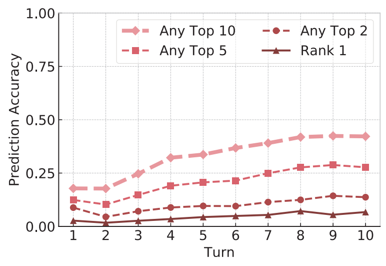


(e) DBSCAN $m_{Pts} = 5$

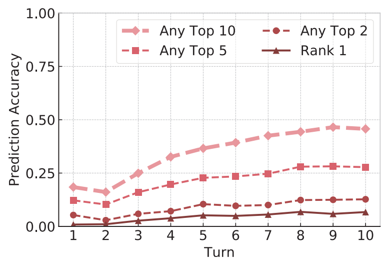


(f) DBSCAN $m_{Pts} = 10$

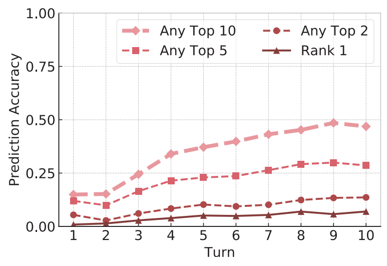
Figure 5 Accuracy for predicting cards that may appear in the remaining turns of the game bucketed by turn. Each model (a)-(f) ranks cards according to their estimated probability of appearance. The validation set is used to determine the accuracy of each prediction for every rank. The highest-ranked card is marked in green, while the prediction of the 10th ranked is shown in red. The average accuracy of all ten ranks is shown in blue.



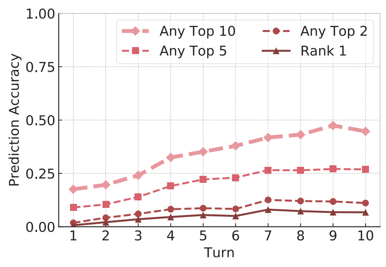
(a) k-means



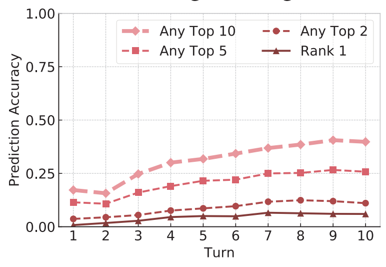
(b) single linkage



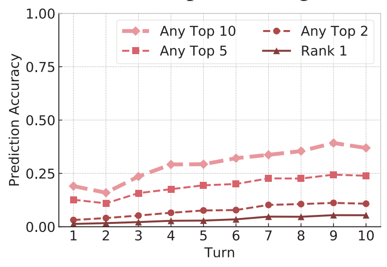
(c) complete linkage



(d) DBSCAN $m_{Pts} = 2$



(e) DBSCAN $m_{Pts} = 5$



(f) DBSCAN $m_{Pts} = 10$

Figure 6 Accuracy for the aggregated prediction of cards that may be played by the opponent during its next turn bucketed by the turn the prediction has been made. Each model (a)-(f) ranks cards according to their estimated probability of appearance. The validation set is used to determine the accuracy in case the ranks 1-k are considered. Values describe the accuracy of the top ranked being correctly predicted.

game Hearthstone. We chose to represent decks in the form of (fuzzy) multisets and define a centroid of clusters of such multisets. On the basis of fuzzy multisets, we clustered real player deck data using multiple clustering algorithms. The result of each algorithm was evaluated in context of real player data and has shown to match human expert player labels of deck archetypes reasonably well. Based on the extracted deck archetypes we proposed a sampling algorithm, which can be used to determinate a state. Predicted state determinizations have been evaluated regarding the prediction accuracy of upcoming cards of the whole game and the opponent's next turn. Results show that the process is able to identify upcoming cards with high accuracy. This forms an excellent basis for applying the proposed approach to state-determinizing search agents and may allow agents to play complex games such as Hearthstone on a higher level than they can currently achieve. Our next step will be to compare state-determinization methods and their effects on an agent's game-playing performance.

While our process currently assumes a static meta-game, we are eager to explore dynamic clustering and prediction schemes in the future. Since web sources frequently update their databases, it may be interesting to capture the dynamic evolvement of new strategies and decks among human players. At the same time, it would allow the agent to keep track of popular strategies to play accordingly. A better understanding of the process in which decks emerge may even help to construct new decks or to further balance the game. However, much work needs to be done until these challenges can be solved.

CONFLICT OF INTEREST

The authors declare they have no conflicts of interest.

AUTHORS' CONTRIBUTIONS

We would not like to explicitely state the authors contributions.

Funding Statement

No funding to be reported.

ACKNOWLEDGMENTS

Nothing to report.

REFERENCES

[1] Blizzard Entertainment, Hearthstone webpage, 2019. https://playhearthstone.com

[2] Statista Inc., Number of hearthstone: heroes of warcraft players worldwide as of November 2018 (in millions), 2019. https://www.statista.com/statistics/323239/number-gamers-hearthstone-heroes-warcraft-worldwide/

[3] Hearthstone Top Decks, Hearthstone beginner's guide 2018 – guides, tips, and tricks for new players!, 2019. https://www.hearthstonetopdecks.com/hearthstone-beginners-guide-2017-guides-tips-tricks-new-players/

[4] A. Dockhorn, S. Mostaghim, Introducing the hearthstone-ai competition, CoRR, abs/1906.04238, 2019.

[5] J.S.B. Choe, J.K. Kim, Enhancing monte carlo tree search for playing hearthstone, in IEEE Conference on Computatonal Intelligence, London, UK, 2019,

[6] A. Dockhorn, M. Frick, U. Akkaya, R. Kruse, Predicting opponent moves for improving hearthstone AI, in: J. Medina, M. Ojeda-Aciego, J.L. Verdegay, D.A. Pelta, I.P. Cabrera, B. Bouchon-Meunier, R.R. Yager (Eds.), 17th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, IPMU 2018, Springer International Publishing, Cham, Switzerland, 2018, pp. 621–632.

[7] J. Long, N.R. Sturtevant, M. Buro, T. Furtak, Understanding the success of perfect information monte carlo sampling in game tree search, in Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, (2010), pp. 134–140.

[8] D. Whitehouse, E. Powley, P. Cowling, Determinization and information set monte carlo tree search for the card game dou di zhu, in Proceedings of the 2011 IEEE Conference on Computational Intelligence and Games (CIG'11), IEEE, Seoul, South Korea, 2011, pp. 87–94.

[9] A. Santos, P.A. Santos, F.S. Melo, Monte carlo tree search experiments in hearthstone, in 2017 IEEE Conference on Computational Intelligence and Games (CIG), IEEE, New York, NY, USA, 2017.

[10] P. García-Sánchez, A. Tonda, A.J. Fernández-Leiva, C. Cotta, Optimizing hearthstone agents using an evolutionary algorithm, Knowl. Based Syst. 188 (2020), 105032.

[11] S. Sievers, M. Helmert, A doppelkopf player based on UCT, in: S. Hölldobler, R. Peñaloza, S.Rudolph (Eds.), Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 9324, Springer, Cham, Switzerland, 2015, pp. 151–165.

[12] A. Dockhorn, C. Doell, M. Hewelt, R. Kruse, A decision heuristic for Monte Carlo tree search doppelkopf agents, in 2017 IEEE Symposium Series on Computational Intelligence (SSCI), IEEE, Honolulu, HI, USA, 2017, pp. 1–8.

[13] E. Bursztein, I am a legend: hacking hearthstone using statistical learning methods, in 2016 IEEE Conference on Computational Intelligence and Games (CIG), IEEE, Santorini, Greece, 2016, pp. 1–8.

[14] A. Dockhorn, T. Schwensfeier, R. Kruse, Fuzzy multiset clustering for metagame analysis, in Proceedings of the 2019 Conference of the International Fuzzy Systems Association and the European Society for Fuzzy Logic and Technology (EUSFLAT 2019), Atlantis Press, Paris, France, 2019.

[15] S. Miyamoto, Fuzzy multisets and their generalizations, in: C. Calude, G. Puaun, G. Rozenberg, A. Salomaa (Eds.), Multiset Processing, Mathematical, Computer Science, and Molecular Computing Points of View [Workshop on Multiset Processing, WMP 2000, Curtea de Arges, Romania, August 21-25, 2000], vol. 2235 of Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, Germany, 2000, pp. 225–236.

[16] R.R. Yager, On the theory of bags, Int. J. Gen. Syst. 13 (1986), 23–37.

[17] C.C. Aggarwal, C.K. Reddy, Data Clustering: Algorithms and Applications, CRC Press, Boca Raton, FL, USA, 2013.

[18] M.R. Berthold, C. Borgelt, F. Höppner, F. Klawonn, Guide to Intelligent Data Analysis. Texts in Computer Science, Springer, London, UK, 2010.

[19] C. Borgelt, R. Kruse, Agglomerative fuzzy clustering, in: M. Ferraro *et al*. (Eds.), International Conference on Soft Methods in Probability and Statistics, Springer, Cham, Switzerland, 2016, pp. 69–77.

[20] R. Kruse, C. Borgelt, C. Braune, S. Mostaghim, M. Steinbrecher, Computational Intelligence, Texts in Computer Science, second ed., Springer, London, UK, 2016.

[21] K. Honda, S. Miyamoto, H. Ichihashi, Hidetomo Ichihashi. Algorithms for Fuzzy Clustering: Methods in C-Means Clustering with Applications, Studies in Fuzziness and Soft Computing 229, first ed., Springer-Verlag, Berlin, Heidelberg, Germany, 2008,

[22] J.B. MacQueen, Some methods for classification and analysis of multivariate observations, in 5th Berkeley Symposium on Mathematical Statistics and Probability, Berkeley, California, USA, 1967, vol. 1, pp. 281–297. https://projecteuclid.org/euclid.bsmsp/1200512992

[23] R. Sibson, SLINK: an optimally efficient algorithm for the single-link cluster method, Comput. J. 16 (1973), 30–34.

[24] D. Defays, An efficient algorithm for a complete link method, Comput. J. 20 (1977), 364–366.

[25] M. Ester, H.P. Kriegel, J. Sander, X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, in Second International Conference on Knowledge Discovery and Data Mining (KDD-96), Portland, Oregon, USA, 1996, pp. 226–231. ,

[26] M. Levandowsky, D. Winter, Distance between sets, Nature. 234 (1971), 34–35.

[27] A. Dockhorn, Project files and supporting material, 2020. https://github.com/ADockhorn/StateDeterminizationExperiments

[28] HearthSim, HSReplay.net, 2019. https://hsreplay.net/decks/

[29] A. Rosenberg, J. Hirschberg, V-measure: aconditional entropy-based external cluster evaluation measure, Comput. Linguistics. 1 (2007), 410–420.

[30] A. Dockhorn, C. Braune, R. Kruse, An alternating optimization approach based on hierarchical adaptations of DBSCAN, in 2015 IEEE Symposium Series on Computational Intelligence, IEEE, Cape Town, South Africa, 2015, pp. 749–755.