Decision-Theoretic Clustering of Strategies

Nolan Bard
University of Alberta
Edmonton, Alberta
nolan@cs.ualberta.ca

Deon Nicholas
University of Waterloo
Waterloo, Ontario
dnichola@uwaterloo.ca

Csaba Szepesvári
University of Alberta
Edmonton, Alberta
szepesva@ualberta.ca

Michael Bowling
University of Alberta
Edmonton, Alberta
bowling@cs.ualberta.ca

Abstract

Clustering agents by their behaviour can be crucial for building effective agent models. Traditional clustering typically aims to group entities together based on a distance metric, where a desirable clustering is one where the entities in a cluster are spatially close together. Instead, one may desire to cluster based on actionability, or the capacity for the clusters to suggest how an agent should respond to maximize their utility with respect to the entities. Segmentation problems examine this decision-theoretic clustering task. Although finding optimal solutions to these problems is computationally hard, greedy-based approximation algorithms exist. However, in settings where the agent has a combinatorially large number of candidate responses whose utilities must be considered, these algorithms are often intractable. In this work, we show that in many cases the utility function can be factored to allow for an efficient greedy algorithm even when there are exponentially large response spaces. We evaluate our technique theoretically, proving approximation bounds, and empirically using extensive-form games by clustering opponent strategies in toy poker games. Our results demonstrate that these techniques yield dramatically improved clusterings compared to a traditional distance-based clustering approach in terms of both subjective quality and utility obtained by responding to the clusters.

1 Introduction

In domains where maximizing an agent's utility is the primary goal, agents may need to model other agents or entities in their environment to improve their utility. Clustering techniques can be beneficial in these settings by allowing the agent to partition the set of entities into similar groups called clusters. In this setting, a clustering's *actionability* — its capacity to suggest responses with high utility to the agent — is the fundamental clustering objective. Contrast this with many traditional clustering problems, such as k-means and k-medians, where similarity is measured by some notion of *spatial* distance between the entities within the same cluster. For instance, the k-means objective is to minimize the within-cluster sum of squared Euclidean distances between each of the entities and their cluster's centroid. Despite an abundance of spatial clustering techniques, these techniques may fail to capture similarity in how the agent should respond to the entities.

Copyright © 2015, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

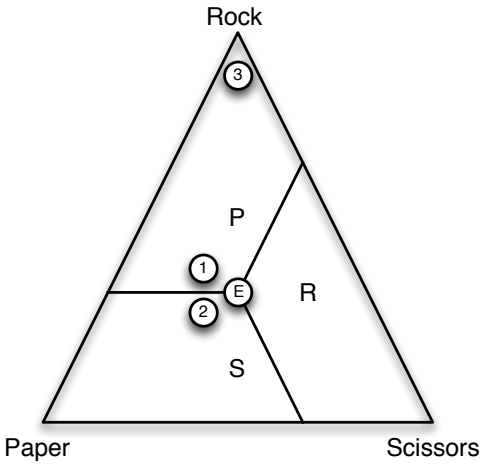


Figure 1: Rock-paper-scissors strategy simplex partitioned into best response regions. Despite the spatial proximity of 1 and 2, they fall in distinct best response regions (P and S).

To illustrate, we briefly examine the game of Rock-Paper-Scissors. In this game, a static agent's behaviour can be specified by their probability distribution over choosing rock, paper, and scissors. Figure 1 depicts a simplex representing the space of possible probability distributions over these three actions. The point E is the game's Nash equilibrium of $1/3$ for each action. Consider the points labelled 1,2, and 3. Although 1 and 2 are spatially close, an agent's **response** for how to act with respect to them should be different as their strategies do not share the same best response. In contrast, 1 and 3 share the same best response (always play paper) despite being spatially distant.

Although this type of decision-theoretic clustering may appear to be a niche problem at first glance, it is actually related to a range of optimization problems. Kleinberg and colleagues (1998) introduced and formalized this style of clustering problem in the data mining community as *segmentation problems*. Their work showed an equivalence between a form of maximum coverage optimization and this type of clustering. Lu and Boutilier (2011) also highlighted

several parallels between their budgeted social choice model and segmentation problems.

Due to the computational complexity of segmentation problems (and many other similar maximum coverage based problems) one typically forgoes exact solutions for efficient approximate solutions to these problems. In particular, greedy approximations related to Nemhauser and colleagues' (1978) greedy algorithm for maximizing monotone submodular functions are commonly used to attack a segmentation problem's maximum coverage formulation. These techniques iteratively construct a solution by evaluating the marginal gain of each feasible response and adding the greedy-best response. Unfortunately, as Kleinberg *et al.* (2004) noted, such greedy algorithms may not be efficient as even a single step can be an NP-complete problem.

In this paper, we examine segmentation problems where the response space is either exponential or infinite and the standard greedy approximation is computationally infeasible. We show that despite this, in certain cases where the utility function can be factored, an efficient response oracle can be constructed. We then use such a response oracle to operate directly on the partitioning formulation of a segmentation problem by greedily merging clusters together in an agglomerative (*i.e.*, "bottom up") clustering algorithm. Finally, we evaluate our technique both theoretically and empirically. Our theoretical results provide a guarantee on the worst case performance of our greedy algorithm relative to the optimal clustering into k sets. This approximation bound is shown to be tight within a factor of 2. We empirically evaluate our technique using extensive-form games by clustering agent strategies in two toy games of poker: Kuhn poker and Leduc hold'em. Our results highlight the benefit of clustering strategies based on their actionability rather than their spatial similarity by contrasting our greedy method with a traditional k-means clustering approach.

2 Background

We begin our exposition by introducing segmentation problems, followed by a brief description of our experimental domain of extensive-form games.

2.1 Segmentation Problems

Segmentation problems examine the challenge of determining how an agent should respond to maximize utility given information about different entities (Kleinberg, Papadimitriou, and Raghavan 1998)[1]. For example, a commercial enterprise with information about their customers could act homogeneously across the customers, but the enterprise may be able to increase its utility by tailoring their **response** (*e.g.*, marketing strategy, product line) to each customer's preferences. While such individual personalization is often impractical, a more limited form of personalization where the market of customers is segmented (*i.e.*, clustered) into k groups may still be beneficial.

One way to view segmentation problems is as clustering problems where the desired clustering depends on a utility function $u(e, r)$ that specifies the utility of **response** $r \in R$ with respect to an **entity** $e \in E$ being clustered. Unlike traditional clustering approaches, this approach directly optimizes for the actionability of the clustering.

[1] Kleinberg and colleagues referred to the entities and responses as customers and decisions, respectively.

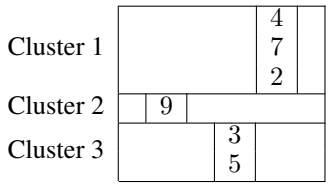


Figure 2: Matrix view of a segmentation problem with an objective value of 30.

More formally, let E, R be sets (E finite), and $u : E \times R \to \mathbb{R}$ a **utility function**. For convenience, we let $\mathrm{Part}(E)$ denote the **set of partitions of E**:

$$\mathrm{Part}(E) = \left\{ P \subset 2^{2^E} : \begin{array}{l} \cup P = E, \\ A, B \in P \Rightarrow A \cap B = \emptyset \end{array} \right\}.$$

For $k \in \mathbb{N}$, we shall also use $\mathrm{Part}_k(E)$ to denote the **set of k-element partitions of E**: $\mathrm{Part}_k(E) = \{P \in \mathrm{Part}(E) : |P| = k\}$.

We abuse notation slightly to define several recurring terms for utility over subsets and partitions of E. For $P \in \mathrm{Part}(E)$, $C \subset E$, $r \in R$, let

$$u(C, r) = \sum_{e \in C} u(e, r),$$
$$u(C) = \max_{r \in R} u(C, r), \text{and}$$
$$u(P) = \sum_{C \in P} u(C).$$

In words, $u(C, r)$ is the total utility of a "cluster" of entities $C \subset E$ when the response is r, $u(C)$ is the utility of C and $u(P)$ is the utility of partition P.

Then the partitioning form of a segmentation problem considers the problem of finding a partition of E that gives the highest utility amongst all k-element partitions:

$$P_k^* = \operatorname*{argmax}_{P \in \mathrm{Part}_k(E)} u(P) = \operatorname*{argmax}_{P \in \mathrm{Part}_k(E)} \sum_{C \in P} \max_{r \in R} \sum_{e \in C} u(e, r). \quad (1)$$

Finally, we let u_k^* denote the **utility of an optimal k-element partition**: $u_k^* = \max_{P i \in \mathrm{Part}_k(E)} u(P)$.

To provide more visual intuition of this problem, we also describe it as an optimization over the following matrix. Let U be an $|E| \times |R|$ matrix where the (i, j)-th entry $U_{i,j} = u(e_i, r_j)$. Then, given $k < |E|$, the goal of the optimization is to find a partition $P = \{C_1, \ldots, C_k\}$ of the rows of U that maximizes the sum of the utilities over the rows when all rows in the same cluster must share the same response column r_j. Figure 2 depicts an example of this matrix form.

Kleinberg and colleagues showed that this partitioning view of a segmentation problem is also equivalent to the following maximum coverage based optimization,

$$\operatorname*{argmax}_{\substack{R' \subseteq R \\ |R'| = k}} \sum_{e \in E} \max_{r \in R'} u(e, r). \quad (2)$$

In our matrix view of this problem, one can think of Equation 2 as choosing the set of k columns of the matrix U that maximally cover the rows of the matrix. The utility of optimal solutions to these two views of a segmentation problem not only have equal value, u_k^*, but it is also straightforward to construct a solution to Equation 2 given a solution to Equation 1, and vice versa.

The equivalence of these two problems means that techniques for solving the maximum coverage based problem in Equation 2 can also be used to solve the partitioning problem in Equation 1. Unfortunately, this type of weighted maximum coverage problem is known to be computationally hard. Cornuejols *et al.* (1977) showed that when u is a nonnegative utility function, the weighted maximum coverage optimization in Equation 2 is NP-complete. Kleinberg *et al.* also showed on several constrained response spaces that for u a general linear function, the segmentation problems are still NP-complete (1998), or more specifically MAXSNP-complete (2004).

Due to these negative complexity results, approximate solutions to segmentation problems are typically sought in lieu of exact solutions. Nemhauser and colleagues (1978) showed that when u is constrained to nonnegative values, the weighted maximum coverage problem is submodular and a $(1 - 1/e)$-approximation to the optimal solution can be computed using a greedy approximation algorithm. Their greedy algorithm iteratively constructs $R' \subseteq R$ by adding $r \in R \setminus R'$ that maximally increases the objective given the previously selected elements in R'. In many natural settings the response space R is either exponential or infinite and solving even a single step of such a greedy algorithm may be NP-complete. Kleinberg *et al.* (2004) introduce several approximation algorithms for segmentation problems. In addition to some greedy approximations, which greedily add the response that maximizes the marginal gain like Nemhauser and colleagues' algorithm, they also introduce efficient approximation algorithms for certain domains that avoid enumerating the response space by exploiting properties of the entities being segmented.

Next, we provide a high-level description of our experimental domain of extensive-form games.

2.2 Extensive-form Games

Extensive-form games provide a general model to represent sequential interactions between agents in an environment. Extensive-form games can be viewed as a tree consisting of nodes corresponding to states of the game where some player acts, and edges representing each of the actions available to a player. Leaves of the tree represent the end of the game and assign utilities to each of the players. In stochastic games, a special chance player acts according to a known distribution. In games of imperfect information, some actions cannot be observed by some of the players. These players are then unable to distinguish which of several game states they are in. A set of these indistinguishable states is called an **information set**. A **strategy** for a player specifies a distribution for each information set over the available actions.

3 Exploiting Structured Utility

In settings where the response space is too large to enumerate efficiently, algorithms that avoid enumerating the candidate responses are necessary. We examine problems where structure in the utility function u allows the best response for a given set of entities to be efficiently computed despite a prohibitively large response space. Next, we formalize this response oracle, provide examples of problems where such an oracle exists, and show how one can incorporate a response oracle into an efficient greedy agglomerative clustering algorithm.

3.1 Response Oracles

In some settings the utility function u can be factored to enable the efficient computation of the response (*i.e.*, column of U) that maximally covers a given set of entities. Formally, given $C \subseteq E$ a **response oracle** $f : 2^E \to R$ is defined as

$$f(C) \equiv \operatorname*{argmax}_{r \in R} u(C, r).$$

By definition, this means $u(C, f(C))$, the utility obtained by the response oracle on C, is $u(C)$.

We provide some domains where such a response oracle can be computed efficiently, taking time logarithmic in the size of the response space, and then present a greedy algorithm that capitalizes on this oracle.

Example: Extensive-form Games.

Let the entities E consist of observed opponent strategies and let the response space R be the possible strategies that our agent could employ. Even if our agent only considers pure strategies (where a player acts deterministically at each of the game's information sets) the size of the response space is exponential in the number of the game's information sets. The response oracle $f(C)$ in this setting is a best response to the average of the sequence-form representations (Koller, Megiddo, and von Stengel 1994) of the strategies in C. Due to an extensive-form game's tree structure, this can be computed in time linear in the number of information sets.

Example: Budgeted Social Choice.

In the case of maximizing social welfare, Lu and Boutilier's (2011) limited choice model of budgeted social choice is equivalent to the segmentation problem optimization in Equation 2 (with entities and responses instead called agents and alternatives, respectively). In their formulation, they assume that the responses can be enumerated and use Nemhauser and colleagues' greedy algorithm. Suppose the responses consist of products represented by feature vectors $r = (r_1, \ldots, r_n) \in R_1 \times \ldots \times R_n$ and that the utility function can be factored into $u(e, r) = \sum_{i=1}^{n} u_i(e, r_i)$ where $u_i : E \times R_i \to \mathbb{R}$. Then the response oracle $f(C)$ can compute each r_i of the optimal response as $\operatorname{argmax}_{r_i \in R_i} \sum_{e \in C} u_i(e, r_i)$. Instead of enumerating the $\prod_{i=1}^{n} |R_i|$ possible responses in R, the response oracle can be computed efficiently in time $O(|C| \sum_{i=1}^{n} |R_i|)$.

3.2 A Greedy Heuristic

Since both exact solutions and algorithms similar to Nemhauser and colleagues' greedy algorithm are infeasible

on segmentation problems with a large response space, we propose an alternative greedy algorithm. Our algorithm is efficient, requiring a polynomial number of oracle queries, provided an efficient response oracle. Although our algorithm also acts greedily based on the marginal change in utility, it does so when considering how to merge clusters in an agglomerative hierarchical clustering algorithm (Ward 1963).

In an agglomerative clustering algorithm, one starts with a partition of singletons and builds a solution in a "bottom up" manner by iteratively coarsening the partition. Unlike Nemhauser and colleagues' greedy algorithm, which acted greedily according to the *maximum marginal gain*, we greedily merge clusters that incur the *minimal marginal loss*. This is because the objective value of Equation 1 is vacuously maximized by the initial partition of singletons (when the size of the partition is unconstrained). One can view this as a greedy heuristic for the following optimization.

$$\min_{P \in \mathrm{Part}_k(E)} \left[\sum_{e \in E} \max_{r^* \in R} u(e, r^*) - u(P) \right] \tag{3}$$

Conceptually, this objective function represents the loss suffered due to clustering entities together as opposed to responding to the entities individually. Note that a partition that optimizes our original objective in Equation 1 also optimizes Equation 3, and vice versa.

Our algorithm starts with the trivial partition $P_0 = \{\{e\} : e \in E\}$ where every entity is in its own set. On each iteration we coarsen the partition by greedily merging together the two sets in the current partition that incur the minimal marginal loss in the objective value of Equation 3. This is repeated until we have a partition that satisfies the cardinality constraint k.

More formally, let $\mathrm{Coarse}(P)$ be all the 1-step coarsenings of partition $P = \{C_1, \ldots, C_k\}$:

$$\mathrm{Coarse}(P) = \{\mathrm{Merge}(P, C_i, C_j) : 1 \leq i < j \leq k\},$$

where

$$\mathrm{Merge}(P, C_i, C_j) = P \setminus \{C_i, C_j\} \cup \{C_i \cup C_j\}.$$

is the partition that results from P by merging C_i and C_j. Then, the greedy algorithm can be written as in Algorithm 1. Note that the combination of sets $C_i, C_j \in P$ that produce the optimal coarsening are exactly those for which the marginal loss $u(C_i) + u(C_j) - u(C_i \cup C_j)$ is minimal.

A naive implementation of this algorithm must compute the marginal loss for all combinations $C_i, C_j \in P$ on each of the $|E| - k$ iterations. This implementation would require $O(|E|^3)$ calls to the oracle. By noticing some structure in the computation, we can use memoization to improve this. After computing the marginal loss for all combinations $C_i, C_j \in P$ of candidate merges on the first iteration, the marginal losses for all candidates can be updated after each merge with at most $O(|E|)$ calls to the response oracle. If C_i and C_j are merged, we need only compute the marginal losses for all pairs of clusters that involve the new cluster $C_i \cup C_j$. This memoization implementation only needs a total of $O(|E|^2)$ calls to the oracle. Additionally, since we know that $u(C_i) + u(C_j) - u(C_i \cup C_j) \geq 0$, we can lazily evaluate the marginal losses (only evaluating them while no zero loss candidate exists) to further reduce computation. Finally, each marginal loss computation is independent from the others and therefore amenable to parallelization.

Algorithm 1 Greedy response oracle clustering

Require: Set E of entities, a response oracle f, utility function u, k
Initialization: $G = P_0 = \{\{e\} : e \in E\}$
while $|G| > k$ **do**
  $G \leftarrow \mathrm{argmax}_{P \in \mathrm{Coarse}(G)} u(P)$
end while
return G

Unlike clustering algorithms where the desired number of clusters k needs to be specified in advance (*e.g.*, Lloyd's algorithm for k-means), our greedy algorithm's iterative and deterministic nature means that it could be run a single time for $|E|$ iterations, reporting the partitions and objective value on each iteration. This enables users to directly evaluate the trade-off between the objective and the number of clusters.

In the remainder of the paper, we evaluate our greedy algorithm both theoretically, proving approximation bounds on the solution quality, and empirically by clustering agent strategies in toy poker games. We begin with our theoretical analysis.

4 Theoretical Results

Our theoretical analysis examines the worst-case behaviour of our greedy clustering algorithm. Theorem 1 establishes a lower bound on the utility of a clustering produced by our greedy algorithm, relative to the optimal clustering into k sets. Theorem 2 then shows that this lower bound is tight (within a factor of 2) and cannot be improved substantially.

In this section we fix E and we denote its cardinality by m. Using the subsequent three lemmas, we prove the following result:

Theorem 1. *Let u be a nonnegative valued utility function, $1 \leq k \leq m$. Then $u(G_k) \geq \max\left(\frac{1}{k}, \frac{k}{m}\right) u_k^* \geq \frac{1}{\sqrt{m}} u_k^*$, where G_k is a k-element partition of E returned by the greedy algorithm.*

Note that the second inequality follows trivially from $\max\left(\frac{1}{k}, \frac{k}{m}\right) \geq \min_{s>0}\left(\frac{1}{s}, \frac{s}{m}\right) = \frac{1}{\sqrt{m}}$. Hence, it remains to prove the first inequality. We prove this by showing that $u(G_k) \geq \frac{k}{m} u_k^*$ in Lemma 2, and that $u(G) \geq \frac{1}{k} u_k^*$ in Lemma 3. We begin by proving a lower bound on the utility for a single coarsening step of the greedy algorithm.

Lemma 1. *Let u be a nonnegative valued utility function, $G \in \mathrm{Part}(E)$, $C = \mathrm{argmin}_{C \in G} u(C)$. Then, for any $C' \in G$, $C' \neq C$, $\max_{P \in \mathrm{Coarse}(G)} u(P) \geq u(\mathrm{Merge}(G, C, C')) \geq \left(1 - \frac{1}{|G|}\right) u(G)$.*

Proof. The first inequality holds by the definition of Coarse. Hence, it remains to prove the second. For this, let $k = |G|$. By the choice of C and the definition of $u(G)$, $u(G) \geq$

$ku(C)$, or $u(C) \leq u(G)/k$. Pick any $C' \in G$, $C' \neq C$ and set $G' = \mathrm{Merge}(G, C, C')$. Let $r' = \mathrm{argmax}_{r \in R} u(C', r)$. Then,

$$\begin{aligned}
u(G') &= \sum_{A \in G'} u(A) \\
&= \overbrace{\sum_{A \in G} u(A)}^{u(G)} + u(C \cup C') - (u(C) + u(C')) \\
&\geq u(G) + u(C \cup C', r') - (u(C) + u(C')) \\
&= u(G) + u(C, r') + \cancel{u(C', r')} - u(C) - \cancel{u(C')} \\
&\geq u(G) - u(C) \qquad \text{(since } u \text{ is nonnegative)} \\
&\geq \left(1 - \frac{1}{k}\right) u(G)\,. \qquad \text{(by the choice of } C\text{)}
\end{aligned}$$

□

Lemma 2. *Assume that u is nonnegative valued. Fix $1 \leq i \leq m$ and let G_i denote the i-element partition returned by the greedy algorithm. Then, $u(G_i) \geq \frac{i}{m} u(G_m) \geq \frac{i}{m} u_i^*$.*

Proof. By using Lemma 1 $(m - i)$-times, we get

$$u(G_i) \geq \left(\prod_{j=i}^{m-1} \frac{j}{j+1} \right) u(G_m)\,,$$

which proves the first half of the lemma. For the second part just combine this with $u(G_m) = u_m^* \geq u_{m-1}^* \geq \cdots \geq u_i^*$. □

It remains to prove that the following lemma holds:

Lemma 3. *Assume that u is nonnegative valued. Let $G \in \mathrm{Part}_k(E)$. Then, $u(G) \geq \frac{1}{k} u_k^*$.*

Proof. Let $P_k^* = \mathrm{argmax}_{P \in \mathrm{Part}_k(E)} u(P)$, and $C = \mathrm{argmax}_{C' \in P_k^*} u(C')$. Then, $u_k^* = u(P_k^*) \leq k u(C)$, hence it suffices to show that $u(G) \geq u(C)$. Let $r_C = \mathrm{argmax}_{r \in R} u(C, r)$. We have

$$\begin{aligned}
u(G) &= \sum_{C' \in G} u(C') \\
&\geq \sum_{C' \in G} u(C', r_C) = \sum_{e \in E} u(e, r_C) \\
&\geq u(C, r_C) \qquad \text{(since } u \text{ is nonnegative)} \\
&= u(C)\,.
\end{aligned}$$

□

The next result establishes that the bound in Theorem 1 cannot be substantially improved:

Theorem 2. *Let k be a positive integer. For each $\varepsilon > 0$, there exists a nonnegative utility function on a set of entities E of $m = k^2$ elements and on a set R of $k^2 + k$ responses such that if G_k is the k-element partition returned by the greedy algorithm when fed with k and u then $u(G_k) - \varepsilon \leq \frac{2}{\sqrt{m}} u_k^*$.*

Proof. We will construct a matrix of size $k^2 \times (k^2 + k)$ holding the values of the utility function. Let $E = \{1, \ldots, k^2\}$, I denote the $k \times k$ identity matrix, $\varepsilon > 0$ and define the $k \times k$ matrix Q_ε by

$$Q_\varepsilon = \begin{pmatrix} 2+\varepsilon & \varepsilon & \ldots & \varepsilon \\ \varepsilon & 2+\varepsilon & \ldots & \varepsilon \\ \vdots & \vdots & \ddots & \vdots \\ \varepsilon & \varepsilon & \ldots & 2+\varepsilon \end{pmatrix}.$$

Finally, as in Section 2.1, the $k^2 \times (k^2 + k)$ matrix representing the utility function u is given by

$$U = \begin{pmatrix} I & Q_\varepsilon & 0 & \ldots & 0 \\ I & 0 & Q_\varepsilon & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ I & 0 & 0 & \ldots & Q_\varepsilon \end{pmatrix}.$$

Let $C_1 = \{1, \ldots, k\}$, $C_2 = \{k+1, \ldots, 2k\}$, …, $C_k = \{k^2 - k + 1, \ldots, k^2\}$. Note that $\{C_1, \ldots, C_k\} \in \mathrm{Part}_k(E)$. We claim that the greedy algorithm returns the k-element partition $G_k = \{C_1, \ldots, C_k\}$.

To show this, let G_i denote the i-element partition computed by the greedy algorithm. We claim that for any $C \in G_i$, $C \subset C_p$ for some $1 \leq p \leq k$ (i.e., G_i is a refinement of G_k). This clearly suffices to prove that $G_k = \{C_1, \ldots, C_k\}$. Since $G_m = \{\{1\}, \ldots, \{k^2\}\}$, the claim holds for $i = m$.

Let us assume that it holds up to $k < i \leq m$. Consider $C, C' \in G_i$. By the induction hypothesis, G_i is a refinement of G_k and therefore $C \subset C_p$ and $C' \subset C_q$ for some $1 \leq p, q \leq k$. Then, $u(C) = 2 + |C|\varepsilon$ and $u(C') = 2 + |C'|\varepsilon$. If C and C' were merged, the marginal loss due to merging is $\ell(C, C') = u(C) + u(C') - u(C \cup C')$. The greedy algorithm merges the two elements of G_i that minimize this loss. If $p = q$ then $u(C \cup C') = 2 + (|C| + |C'|)\varepsilon$ and $\ell(C, C') = 2$. If $p \neq q$ then $u(C \cup C') = 2 + \max(|C|, |C'|)\varepsilon$ and $\ell(C, C') = 2 + (|C| + |C'| - \max(|C|, |C'|))\varepsilon$. Since $\varepsilon > 0$, the loss due to merging two clusters within the same block is always smaller than that of when the two clusters belong to different blocks. Hence, the greedy algorithm will choose to merge clusters within the same block. This shows that the induction hypothesis also holds for $i - 1$, finishing the proof of the claim.

We also claim that as long as ε is small enough, the optimal k-element partition is $P_k^* = \{\{1, k+1, 2k+1, \ldots, k^2 - k + 1\}, \{2, k+2, 2k+2, \ldots, k^2 - k + 2\}, \ldots, \{2k, 3k, \ldots, k^2\}\}$. Let C_i^* be the ith cluster in P_k^*. Then, for any $C \subset E$, if $n_i = |C \cap C_i^*|$, $1 \leq i \leq k$, $n(C) = \max(n_1, \ldots, n_k)$, we have $n(C) \leq k$ and $u(C) = \max(n_1, \ldots, n_k, 2 + n_1\varepsilon, \ldots, 2 + n_k\varepsilon) = \max(n(C), 2 + n(C)\varepsilon) \leq \max(k, 2 + k\varepsilon) \leq k$, as long as $\varepsilon \leq 1 - \frac{2}{k}$. Hence, for such an ε, for any k-element partition, $P = \{A_1, \ldots, A_k\}$, $u(P) = \sum_{i=1}^k u(A_i) = \sum_{i=1}^k \max(n(A_i), 2 + n(A_i)\varepsilon) \leq k^2 = u(P_k^*)$, showing that P_k^* is indeed an optimal partition.

Now, $u(G_k) = k(2 + k\varepsilon) = 2k + k^2\varepsilon$, while $u_k^* = u(P_k^*) = k^2$. Hence, $\frac{2}{\sqrt{m}} u_k^* = \frac{2}{k} k^2 = 2k \geq u(G_k) - k^2\varepsilon$, finishing the proof. □

While these theoretical results provide guarantees about our greedy algorithm's *worse-case* performance, it can perform much better in practice. In the next section, we empirically evaluate our greedy algorithm's practical performance by clustering agent strategies in two toy poker games.

5 Empirical Results

We demonstrate the performance of our greedy heuristic algorithm by clustering agent strategies for extensive-form games. In particular, our experiments examine poker: a popular family of games that can be modelled as extensive-form games. We begin with a brief introduction of poker and the two small poker games used in our experiments: Kuhn poker and Leduc hold'em. We then describe how we cast this problem as a segmentation problem, the design of our empirical evaluation, and finally present our results.

5.1 Poker

Poker is a family of stochastic imperfect information games where each agent's goal is to maximize their winnings against some opponents. While there are numerous poker variants, their rules share similar structure. A game begins with chance dealing cards from a deck to each player, and typically one or more players are forced to place bets. All bets contribute to the pot which is paid to the winner. The game proceeds through a number of betting rounds consisting of actions by the players. Players can fold (conceding the pot), call (matching the maximum previous bet), or raise (increasing the maximum bet). A betting round ends when all players have matched the last bet, or all but one player folds. If only one player remains, they win the pot and the game ends. Between betting rounds, players' hands change in some way due to chance events. One common way this happens is that chance deals some number of "community" cards which are visible and usable by all players. At the end of the final betting round, if more than one player remains, a showdown occurs. In a showdown, all remaining players reveal their cards, evaluate their hands based on their cards and any community cards, and the player with the strongest hand is awarded the pot (or it is split in the event of a tie). In limit poker variants, bets are of a fixed size and the maximum number of bets within a betting round is limited.

Kuhn Poker Kuhn poker is a toy variant of poker that is small enough that game theoretic analysis of it is tractable and exists (Kuhn 1950). It is a two-player zero-sum poker game with a deck of three cards: jack, queen, and king. In Kuhn poker, both players must make an initial forced bet (ante) and are dealt a single private card. Betting occurs in a single betting round with betting limited to at most a single bet of a fixed size.

In his analysis, Kuhn showed that strategies playing some of the actions were dominated. For example, when holding the king (the strongest card) a player should never fold. If all such actions are eliminated, then strategies in the resulting undominated version of Kuhn poker can be parameterized with three parameters (α, β, γ) for player one, and two parameters (η, ξ) for player two. Our experiments examine this undominated Kuhn poker game.

Leduc Hold'em Leduc hold'em (Southey et al. 2005) is another two-player zero-sum variant of poker which, though larger than Kuhn poker, is still small relative to common poker games played by humans. As in Kuhn poker, the game begins with both players forced to bet an ante and being dealt a single private card. The deck in Leduc hold'em consists of six cards with three ranks (jack, queen, and king) and two suits. Leduc has two betting rounds with betting limited to a maximum of two fixed-size bets per round. After the first betting round, chance deals a public community card. In a showdown, holding the king is no longer the best hand as a pair of cards beats any other hand.

5.2 Experimental Design

To evaluate our greedy decision-theoretic clustering algorithm, we contrast its performance with a k-means clustering algorithm in the domains of Kuhn and Leduc hold'em poker. We start by describing how clustering agents in an extensive-form game can be cast as a segmentation problem, and then detail our benchmark k-means algorithm before moving on to our empirical data.

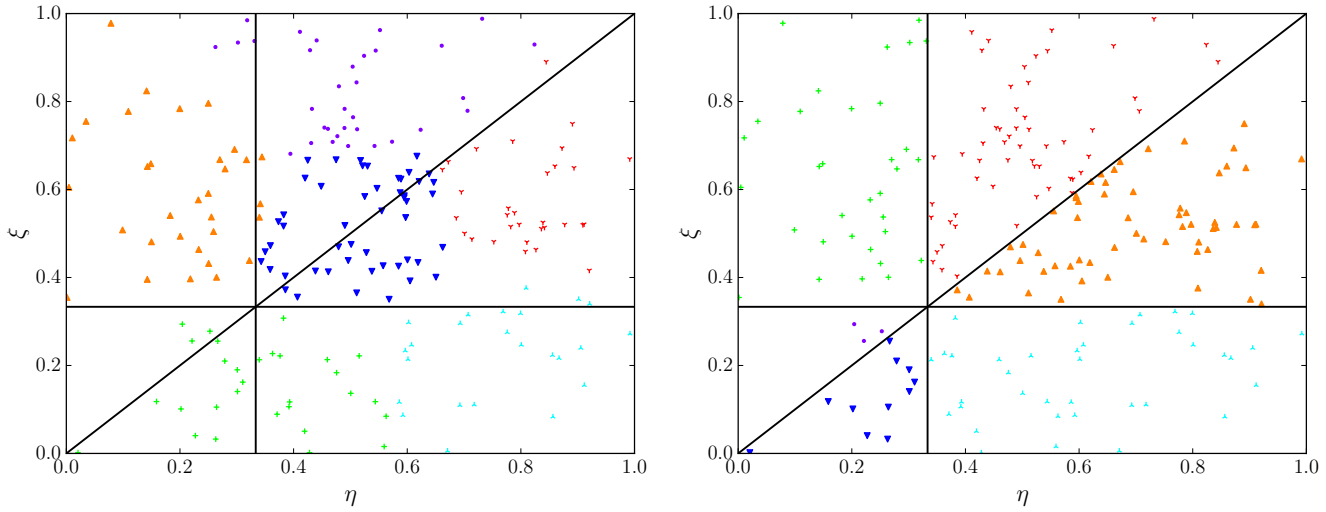
In this setting, we seek a partition of a set E of static agent strategies that optimizes Equation 1. In our experiments, we generate the agents in E by sampling 200 strategies uniformly at random from the strategy space. Though not constructed explicitly, the utility matrix U can be viewed as having a column for each possible strategy and entries $u(e, r)$ corresponding to the expected utility of playing strategy r against static agent e. As mentioned previously, the response oracle $f(C)$ for a set of strategies C is the best response to the average of the sequence-form representations of the strategies in C.

Each of our experiments contrasts our greedy algorithm with the standard k-means clustering algorithm, *i.e.* Lloyd's algorithm (Lloyd 1982), using the sequence-form representation (Koller, Megiddo, and von Stengel 1994) of an agent's strategy as its feature vector. We initialize the cluster centroids using the k-means++ algorithm (Arthur and Vassilvitskii 2007). Note that while the locally optimal clustering found by Lloyd's algorithm may be arbitrarily bad in terms of the k-means objective, initializing with k-means++ provides an approximation guarantee on the solution quality. Despite this, the stochasticity of the k-means initialization impacts which local optimum is found. In our experiments, k-means is restarted 50 times and the clustering with the best k-means objective (*i.e.*, minimal within-cluster sum of squared Euclidean distances) is reported.

5.3 Results

We begin the analysis of our greedy clustering algorithm by examining both its qualitative and quantitative performance when clustering player two's strategies in undominated Kuhn poker.

Kuhn Poker Figure 3 contrasts clusterings produced by k-means and our greedy algorithm to demonstrate the qualitative differences in these clustering approaches. These figures visualize each of the 200 agent strategies according to their two parameters, η and ξ. The marker shape and colour



(a) k-means

(b) Greedy agglomerative clustering

Figure 3: Clusters of player two Kuhn poker agents found using k-means clustering over the sequence-form representations and our greedy heuristic algorithm ($k = 6$). Marker shape and colour indicate cluster membership.

correspond to which cluster each strategy is assigned to. As with the rock-paper-scissors example, the boundary lines between each of the best response regions are plotted. This partitions the strategy space into six regions each with a distinct best response. The k-means and greedy algorithms were run with $k = 6$ clusters as this is sufficient to optimally partition the strategies. Figure 3a illustrates how k-means tends to produce relatively spherical clusters of similar size. Unsurprisingly, k-means' optimization of spatial distances results in clusters that fail to respect these boundaries. In contrast, our greedy algorithm (Figure 3b) is able to exactly partition the agents according to the best response regions.

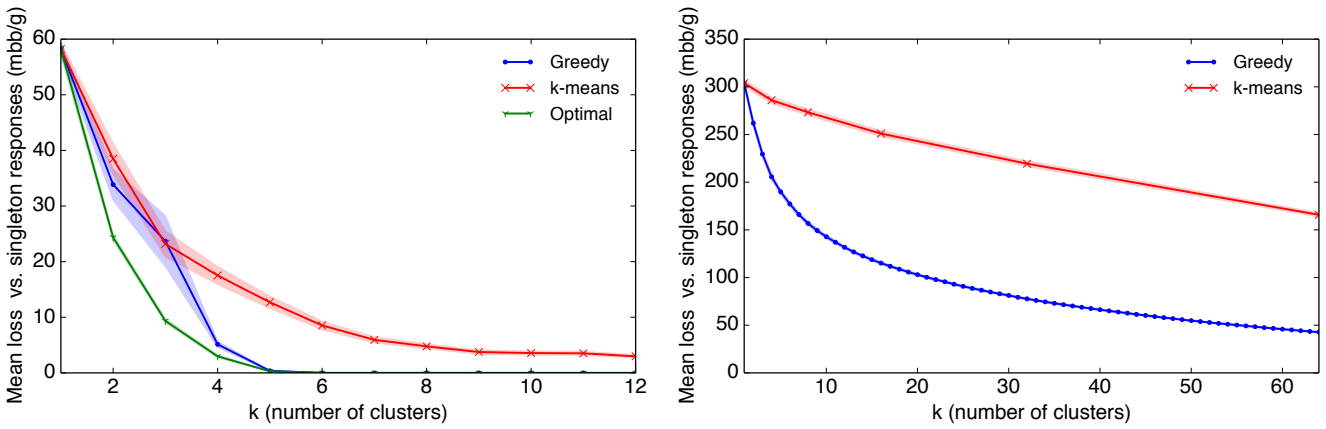
Next, we examine the quantitative performance of these algorithms in terms of the utility lost due to responding to agents in clusters rather than as individuals (as in Equation 3). Figure 4a shows the loss incurred by k-means and our greedy algorithm as we vary the number of clusters for undominated Kuhn poker. In this domain we also compute the optimal clustering through enumerating all possible combinations of k of the six best responses and then inducing the corresponding partition. Results have been averaged over 50 trials each sampling a new set of 200 agents. The trend lines show the mean value for the loss and the surrounding shaded region indicates the 95% confidence interval (which is occasionally difficult to see as it is smaller than the line width). Values are in milli big blinds per game (*i.e.*, thousandths of the initial ante). While the greedy algorithm manages to achieve zero loss once allowed the six clusters required to guarantee the points can be properly partitioned, k-means is unable to reach zero loss even after being given twice as many clusters. This result also highlights how our greedy algorithm can obtain considerably more of the optimal clustering's utility than is guaranteed by our worst-case approximation bound.

Leduc Hold'em Though Kuhn poker provides a convenient domain for visualizing strategies and best response regions, clustering agents in such small domains can be done through direct analysis of the game or brute force enumeration of a player's pure strategies. Leduc hold'em better demonstrates the value of a response oracle as the game is sufficiently large that such enumeration is infeasible. Figure 4b shows similar quantitative performance results for the domain of Leduc hold'em where strategies for player one are being clustered. Note that the results in Figure 4 exploit the fact that the greedy algorithm's performance can be computed for each k with little additional computation (though Figure 4b omits values over 64). It is clear in these results that our greedy clustering algorithm substantially outperforms k-means. In particular, the greedy algorithm achieves approximately the same performance with 7 clusters as k-means does with 64.

Finally, it is interesting to note the rate of improvement as we allow for more clusters. In particular, although the greedy algorithm initially improves rapidly as we increase the number of clusters, the rate of improvement quickly levels off and leaves a very long tail compared to Kuhn poker. This is likely due both to the increased complexity of the game and also the uniform random sampling of the strategy space. Unlike Kuhn poker where we would expect each best response region to contain multiple samples of the 200 strategies, the more complex strategy space of Leduc hold'em likely means any given best response region is (at best) sparsely sampled. If the agents being clustered were covered by relatively few of a game's best response regions, then this long tail may not be present.

6 Conclusion

Agents seeking to maximize their utility may be able to improve their performance by exploiting models of other



(a) Kuhn poker

(b) Leduc hold'em

Figure 4: Performance of different clustering techniques in small poker domains

agents or entities in their environment. In these settings, clustering techniques can be beneficial for extracting similar groups of entities. Despite the ubiquity of spatial clustering techniques, spatial similarity may be insufficient for capturing similarity in how an agent should respond to these groups. Although work on segmentation problems provide techniques to optimize for actionable clusters, these techniques may be computationally infeasible for domains with large response spaces. We introduce an efficient greedy algorithm for this type of decision-theoretic clustering that can exploit the structure of certain domains. We prove worst case approximation bounds on the quality of solutions produced by our greedy algorithm. Finally, we show how to apply this technique to extensive-form games, and empirically demonstrate the value of this approach by comparing it to k-means for clustering agent behaviours in two toy games of poker.

Acknowledgements

The authors would like to thank all of the members of the Computer Poker Research Group at the University of Alberta for helpful conversations pertaining to this research. This research was supported by NSERC and Alberta Innovates Technology Futures. Computing resources were provided by Westgrid, Calcul Québec, and Compute Canada.

References

Arthur, D., and Vassilvitskii, S. 2007. K-means++: The advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, 1027–1035.

Cornuejols, G.; Fisher, M. L.; and Nemhauser, G. L. 1977. Location of Bank Accounts to Optimize Float: An Analytic Study of Exact and Approximate Algorithms. *Management science* 23(8):789–810.

Kleinberg, J. M.; Papadimitriou, C. H.; and Raghavan, P. 1998. A microeconomic view of data mining. *Data Mining and Knowledge Discovery* 2(4):311–324.

Kleinberg, J. M.; Papadimitriou, C. H.; and Raghavan, P. 2004. Segmentation problems. *Journal of the ACM* 51(2):263–280.

Koller, D.; Megiddo, N.; and von Stengel, B. 1994. Fast algorithms for finding randomized strategies in game trees. In *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing*, STOC '94, 750–759.

Kuhn, H. W. 1950. A simplified two-person poker. *Contributions to the Theory of Games* 1:97–103.

Lloyd, S. P. 1982. Least squares quantization in PCM. *IEEE Transactions on Information Theory* 28(2):129–137.

Lu, T., and Boutilier, C. 2011. Budgeted social choice: From consensus to personalized decision making. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*, IJCAI '11, 280–286. AAAI Press.

Nemhauser, G.; Wolsey, L.; and Fisher, M. 1978. An analysis of approximations for maximizing submodular set functions - I. *Mathematical Programming* 14(1):265–294.

Southey, F.; Bowling, M.; Larson, B.; Piccione, C.; Burch, N.; Billings, D.; and Rayner, C. 2005. Bayes bluff: Opponent modelling in poker. In *Proceedings of the 21st Annual Conference on Uncertainty in Artificial Intelligence*, UAI '05, 550–558.

Ward, J. H. 1963. Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association* 58(301):236–244.