Segmentation Problems

Jon Kleinberg * Christos Papadimitriou† Prabhakar Raghavan‡

Abstract

We introduce and study a novel genre of optimization problems, which we call *segmentation problems*. Our motivation, in part, is the development of a framework for evaluating certain data mining and clustering operations in terms of their utility in decision-making. For any classical optimization problem, the corresponding segmentation problem seeks to partition a set of cost vectors into several *segments*, so that the overall cost is optimized. This framework contains a number of standard combinatorial clustering problems as special cases, and many segmentation problems turn out to be MAXSNP-complete even when the corresponding "unsegmented" version is easy to solve. We develop approximation algorithms for two natural and interesting problems in this class — the HYPERCUBE SEGMENTATION PROBLEM and the CATALOG SEGMENTATION PROBLEM — and present a general greedy scheme, which can be specialized to approximate a large class of segmentation problems. Finally, we indicate some connections to local search, game theory, and sensitivity analysis in linear programming.

1 Motivation: a microeconomic view of data mining

Data mining is about *extracting interesting patterns from raw data.* There is some agreement in the literature on what qualifies as a "pattern" (association rules and correlations [1, 7, 14] as well as clustering of the data points [8], are some common classes of patterns sought), but only disjointed discussion of what "interesting" means. Most work on data mining studies how patterns are to be extracted automatically, presumably for subsequent human evaluation of the extent in which they are interesting. Automatically focusing on the "interesting" patterns has received very limited formal treatment. Patterns are often deemed "interesting" on the basis of their *confidence and support* [1], *information content* [24], and *unexpectedness* [17, 23]. The more promising concept of *actionability* — the ability of the pattern to suggest concrete and profitable action by the decision-makers [18, 21, 23] — has not been defined rigorously or elaborated on in the data mining literature.

We want to develop a theory of the value of extracted patterns. We believe that the question can only be addressed in a *microeconomic* framework. *A pattern in the data is interesting only to the extent in which it can be used in the decision-making process of the enterprise to increase utility.*[1] Any enterprise faces an optimization problem, which can generally be stated as

$$\max_{x \in \mathcal{D}} f(x),$$

where $\mathcal{D}$ is the domain of all possible decisions (production plans, marketing strategies, etc.), and $f(x)$ is the *utility* or *value* of decision $x \in \mathcal{D}$. Such optimization problems are the object of study in optimization and microeconomics.[2]

The feasible region $\mathcal{D}$ and the objective $f(x)$ are both comparably complex components of the problem —and classical optimization theory often treats them in a unified way via Lagrange multipliers and penalty functions [3]. However, from our point of view there

*Department of Computer Science, Cornell University, Ithaca NY 14853. Email: kleinber@cs.cornell.edu. Supported in part by an Alfred P. Sloan Research Fellowship and by NSF Faculty Early Career Development Award CCR-9701399. This work was performed in part while visiting the IBM Almaden Research Center.

†Computer Science Division, Soda Hall, UC Berkeley, CA 94720. christos@cs.berkeley.edu. Research performed in part while visiting the IBM Almaden Research Center, and supported by NSF grants CCR-9626361 and IRI-9712131.

‡IBM Almaden Research Center, 650 Harry Road, San Jose CA 95120. pragh@almaden.ibm.com

[1] To quote [6], *"merely finding the patterns is not enough. You must be able to respond to the patterns, to act on them, ultimately turning the data into information, the information into action, and the action into value."*

[2] There is such an optimization problem associated with virtually every enterprise; however *in real life such problems are so involved and complex, that often nobody knows exactly their detailed formulation.* The decision-makers of the enterprise base their decisions on a very rough, approximate, and heuristic understanding of the nature and behavior of the underlying optimization problem. The fact that the details of the optimization problem being solved are nebulous and unknown to the decision-makers does not make the problem less real —or its mathematical study less useful. In fact, economic theory during this century has flourished on models such as these, in which the precise nature of the functions involved is essentially unknowable; the mathematical insights derived from the abstract problem are still valuable as heuristic guides in decision-making.

is a major difference between the two: the feasible region $\mathcal{D}$ is basically endogenous to the enterprise, while the objective function $f(x)$ is a function that reflects the enterprise's interaction with a multitude of other agents in the market (customers, suppliers, employees, competitors, the rest of the world). That is, at a first approximation the objective function can be rewritten as

$$f(x) = \sum_{i \in \mathcal{C}} f_i(x),$$

where $\mathcal{C}$ is a set of agents or other factors influencing the utility of the enterprise. We shall be calling elements of $\mathcal{C}$ "customers." We shall be deliberately vague on what they are. There are two different possible views here: On a concrete level, we can think of them as profiles of customers and other relevant agents, about whom we have gathered relevant information by a first stage of data mining; it is this first stage that our point of view seeks to influence and direct. A more abstract, but potentially equally useful, point of view is that, alternatively, we can also think of the elements of $\mathcal{C}$ as *rows of the raw table being mined* —customers, transactions, shipments, and so on.

What makes this relevant to data mining is the following crucial assumption: we assume that the contribution of customer i to the utility of the enterprise under decision x, $f_i(x)$, is in fact a complicated function of the data we have on customer i. Let y_i denote the data we have on customer i; then $f_i(x)$ is just $g(x, y_i)$, some fixed function of the decision and the data. Hence, our problem is to

$$\max_{x \in \mathcal{D}} \sum_{i \in \mathcal{C}} g(x, y_i).$$

The conventional practice in studying such problems is to replace $\sum_{i \in \mathcal{C}} g(x, y_i)$ by $g(x, \hat{y})$, where $\hat{y}$ is some *aggregate* value[3] (aggregate demand of a product, aggregate consumer utility function, etc.). Such aggregation is understood to be inaccurate, resulting in suboptimal decisions, because of *non-linearities* (non-zero second partial derivatives) in the function $g(x, y_i)$. Aggregation has been tolerated in traditional microeconomics because (1) the computational requirements otherwise would be enormous, and (2) it is difficult to obtain the data y_i. The point in data mining, in our view, is that *we now have the computational power and wealth of data necessary to attack the unaggregated optimization problem, to study the intricate ways in which correlations and clusters in the data affect the enterprise's optimal decisions.*

[3] We use "aggregate" in its microeconomics usage — summary of a parameter over a large population — which is related but not identical to its technical meaning in databases.

Segmentation Problems.

For most of the paper, we consider the class of *segmentation problems*, which are aimed at addressing a particular type of aggregation in optimization problems. As above, we consider a company that has information about a set $\mathcal{C}$ of customers; and we view $\mathcal{D}$ as a set of possible marketing strategies. Any given marketing strategy $x \in \mathcal{D}$ will attract certain customers and fail to attract (or even repel) others. Thus, for each customer $i \in \mathcal{C}$, the utility of the marketing strategy x *with respect to* i can be written as a quantity $f_i(x)$; then, the overall utility $f(x)$ of marketing strategy x can be approximated by the superposition $\sum_{i \in \mathcal{C}} f_i(x)$.

There is a spectrum of degrees of aggregation, between the following two extremes. At one extreme — no aggregation — the enterprise could consider each of the functions f_i separately, and implement $|\mathcal{C}|$ different decisions $x_1, \ldots, x_{|\mathcal{C}|}$, targeting x_i specifically at $i \in \mathcal{C}$. This is clearly not practically feasible for a variety of reasons. The computational effort required to determine this many separate strategies, and the cost to implement them in this extremely targeted way, is prohibitive; moreover, one's estimates of the individual functions f_i are not nearly accurate enough even to make this a meaningful activity. At the other extreme — complete aggregation — the enterprise could compute a single decision x that maximizes $\sum_{i \in \mathcal{C}} f_i(x)$. But this will typically miss certain obvious — and profitable — segmentations of the underlying customer data.

For example, telephone companies in the U.S.A. have divided their customers into two segments: residence and business customers; they offer different terms and prices to the two. Mail-order companies produce several hundred different catalogs each year, targeting only a small number of these at each of the customers on their mailing list. How can such segmentation decisions be arrived at in a principled and automatic manner? In each situation, what is the optimal level of aggregation, and what is the corresponding optimum ensemble of decisions? Segmentation problems, as defined and studied in this paper, are stylized computational problems whose intention is to capture these important questions.

Segmentation problems also relate to *clustering*, an important, classical, and challenging algorithmic problem area [15], of interest in data mining —which it predates. Suppose that a large set of points in a multidimensional space must be partitioned into clusters. How is the quality of such a partition to be judged? There are numerous general-purpose criteria (minimizing the sum of the radii of the clusters, maximizing their distance, maximizing the weight of the edges cut, optimizing information-theoretic criteria, among many more [5, 9, 12, 13, 16]) but very little problem-independent guidance on how to select the most appropriate one. Segmentation problems use explicitly the objective function in the underlying decision-making problem as the clustering criterion. This captures in a very concrete sense one of the main motivations of clustering — namely, flexible decision-making at the cluster level.

Organization of the Paper. In Section 2, we carefully define the class of *segmentation problems* considered here, exploring several formulations of the problem (some of them equivalent) and developing basic facts about their complexity. Sections 3 and 4 are concerned with approximation algorithms for two natural and simply stated segmentation problems: the CATALOG SEGMENTATION PROBLEM and the HYPERCUBE SEGMENTATION PROBLEM. Section 5 develops a connection between segmentation problems and a generalization of submodular functions, which leads to a general greedy strategy for approximating a large class of submodular functions. Finally, in Section 6, we discuss further results that indicate a number of interesting directions for research:

- For segmentation problems that arise from an underlying linear program, we propose a natural local search heuristic that we term the *k-fold simplex method.*
- The above motivation is expressed entirely in terms of the utility function for a *single* enterprise. But one can develop a notion of *segmentation* in a competitive environment, and this leads to the formulation of *segmented matrix games.*
- We also investigate a type of problem involving aggregation and non-linearity that falls apparently outside the framework of segmentation problems — it suggests another sense in which associations and correlations between attributes in a dataset can be deemed "interesting" if they correspond to non-linear terms in the objective function of the enterprise, and has connections with sensitivity analysis in linear programming.

2 Formulating Segmentation Problems

To cast the definitions to come in a concrete setting, we begin with an example. The classical *knapsack problem* asks: given d items each having a *weight* and a *value*, and a bound on the total allowable weight r, select a subset of the items of maximum value with total weight not exceeding r. Here is an application of this problem: suppose that we have a set of *items* to offer for sale to n customers. We are given, for each customer, the subset of items the customer is known to like. We wish to create a *catalog* with r of the items to send to the customers; our objective is to maximize the sum, over these r items, of the number of customers who like each item. This is a special case of the knapsack problem in which each item has unit weight; the (rather trivial) solution is simply to select the r most popular items.

Now suppose, instead, that we are allowed to create *two* catalogs each with r items, sending one of the two to each consumer; obviously, there are cases in which the value we obtain from a pair of catalogs can greatly exceed the value obtainable from one.

CATALOG SEGMENTATION. Given a ground set U and n subsets $S_1, \ldots, S_n$ of U, find two subsets X and Y of U, each of size r, so that

$$\sum_{i=1}^{n} \max(|S_i \cap X|, |S_i \cap Y|)$$

is maximized.

Thus, beginning with a standard optimization problem (an easily solvable version of KNAPSACK), together with a large set of different cost functions, we obtain a *segmented* optimization problem in which the goal is to produce two feasible solutions, evaluating them with respect to a partition of the cost functions.

In general, we are given a domain $\mathcal{D} \subseteq \mathbf{R}^d$ of possible *decisions*, and we are given a set of n *customers,* represented by functions $f_1, \ldots, f_n$ with $f_i : \mathcal{D} \rightarrow \mathbf{R}$. In our formulations, these functions will have a very simple form; typically, $f_i(x) = v_i \cdot x$ for a vector $v_i \in \mathbf{R}^d$. Now, if $\max_{x \in \mathcal{D}} f(x)$ is any optimization problem, its corresponding *segmentation problem* is the following:

SEGMENTATION PROBLEM FOR $(\mathcal{D}, f)$: Given n functions $f_1, \ldots, f_n$ and an integer k, find k solutions $x_1, \ldots, x_k \in \mathcal{D}$ such that the sum

$$\sum_{i=1}^{n} \max_{1 \leq j \leq k} f_i(x_j)$$

is maximized.

Thus, for example, if $\mathcal{D}$ is the set of spanning trees of a graph G, and customer i has a vector of edge weights v_i, then the SPANNING TREE SEGMENTATION PROBLEM would be to produce k spanning trees of G; each customer then chooses the best among these k trees with respect to her edge weights, and the value of the overall solution is determined by summing over all customers. One can define "segmented" versions of essentially any other standard optimization problem in a similar way; note that the definition makes sense in the context of minimization problems as well. In this paper we will be focusing on certain natural segmentation problems that capture the marketing motivation.

We can define another version of the segmentation problem:

SEGMENTATION PROBLEM FOR $(\mathcal{D}, f)$ (PARTITION VERSION): Given n functions $f_1, \ldots, f_n$ and an integer k, find a partition of $\{1, \ldots, n\}$ into k sets $S_1, \ldots, S_k$ such that the sum

$$\sum_{j=1}^{k} \left[\max_{x \in \mathcal{D}} \sum_{i \in S_j} f_i(x) \right]$$

is maximized.

It is easy to see that the two variants are equivalent, essentially because the two max operators commute. The algorithmic significance of this equivalence is that the naive algorithm for solving segmentation problems need not be of complexity $|\mathcal{D}|2^{nk}$ (list all partitions), where $|\mathcal{D}|$ is the number of extreme points of $\mathcal{D}$, but "only" $|\mathcal{D}|^k n$ (list all k-tuples of solutions) — in other words, it is fixed-parameter tractable if we consider $\mathcal{D}$ fixed and n as the truly unbounded parameter. There is another, equally natural, version, in which k is not fixed a priori, but there is a cost γ for each additional segment:

SEGMENTATION PROBLEM FOR $(\mathcal{D}, f)$ (VARIABLE k VERSION): Given n functions $f_1, \ldots, f_n$ and an integer γ, find an integer k, and k solutions $x_1, \ldots, x_k \in \mathcal{D}$ to maximize the sum

$$\left(\sum_{i=1}^{n} \max_{1 \le j \le k} f_i(x_j)\right) - \gamma k.$$

Note the apparent similarity between segmentation problems and *facility location problems*, in which one must "open" some number of facilities to serve customers: there is a cost for each facility opened, and a penalty for each customer-facility distance. The issues in our algorithms here turn out to be technically quite distinct from those in facility location problems. First, our problems center around maximization rather than minimization, and this changes the nature of the approximation questions completely. Moreover, our space of possible decisions is typically exponential or infinite, and only implicitly represented. In recent approximation algorithms for the discrete facility location problem and its variants, on the other hand (see e.g. [22]), the facilities must typically be sited at demand locations, yielding, immediately, a relatively small space of possible decisions.

Complexity

Even the most trivial optimization problems (e.g., maximizing a linear function over the d-dimensional ball, whose ordinary version can be solved by aligning the solution with the cost vector) become NP-complete in their segmentation versions. We summarize the complexity results below:

Theorem 2.1 *The segmentation problems (all three versions) corresponding to the following feasible sets $\mathcal{D}$ are NP-complete: (1) The d-dimensional unit ball, even with $k = 2$; (2) the d-dimensional unit L_1 ball; (3) the r-slice of the d-dimensional hypercube (the* CATALOG SEGMENTATION PROBLEM*), even with $k = 2$; (4) the d-dimensional hypercube, even with $k = 2$; (5) the set of all spanning trees of a graph G, even with $k = 2$.*

Proof Sketch. Notice that the optimization problems underlying these problems are extremely easy: The one underlying (1) can be solved by aligning the solution with the cost vector, the one for (2) has only $2d$ vertices, the one for (3) can be solved by choosing the r most popular elements, and the one for (4) by simply picking the vertex that coordinate-wise agrees in sign with the cost vector. Since (2) has $2d$ vertices it can be solved in $O((2d)^k n)$ time, which is polynomial when k is bounded.

The NP-completeness reductions are surprisingly diverse: (1) is proved by a reduction from MAX CUT, (2) from HITTING SET, (3) from BIPARTITE CLIQUE, and (4) from MAXIMUM SATISFIABILITY with clauses that are equations modulo two. Finally, for SPANNING TREE SEGMENTATION we use a reduction from HYPERCUBE SEGMENTATION (the latter problem is essentially a special case of the former, in which the graph is a path with two parallel edges between each pair of consecutive nodes).

Here we sketch only the proof of (1). Suppose that we have a graph $G = (V, E)$; direct its edges arbitrarily, and consider the node-edge incident matrix of G (the $|V| \times |E|$ matrix with the $(i,j)^{\text{th}}$ entry equal to 1 if the j^{th} edge enters the i^{th} node, -1 if the j^{th} edge leaves the i^{th} node, and 0 otherwise). Let the $|V|$ rows of this matrix define the cost vectors $\{v_1, \ldots, v_n\}$ of the segmentation problem. Thus, we seek to divide these $|V|$ vectors into two sets, S_1 and S_2, and choose an optimal solution for each set. Let $\sigma_i = \sum_{v_j \in S_i} v_j$. Since $\mathcal{D}$ is the unit ball, an optimal solution for S_i is simply the unit vector in the direction of σ_i, and hence the value of the solution associated with (S_1, S_2) is simply the sum of the Euclidean norms, $\|\sigma_1\| + \|\sigma_2\|$. However, it is easy to see that for any partition (S_1, S_2) of the vertices, $\|\sigma_1\| + \|\sigma_2\|$ is twice the square root of the number of edges in the cut (S_1, S_2) (because in the two sums the only entries that do not cancel out are the ones that correspond to edges in the cut); hence, solving the segmentation problem is the same as finding the maximum cut of the graph. ■

When the problem dimension is fixed, most of these problems be solved in polynomial time:

Theorem 2.2 *Segmentation problems (2–5) in the previous theorem can be solved in linear time when the number of dimensions is fixed. Problem (1) (the unit ball) can be solved in time $O(n^2 k)$ in two dimensions, and is NP-complete (for variable k) in three dimensions.*

Proof Sketch. When the number of dimensions is a fixed constant d, the number of extreme solutions in each problem (2–5) is constant ($2d$, $\binom{d}{r}$, 2^d, and d^{d-2}, respectively). Thus the number of all possible sets of k solutions is also a bounded constant, call it c; obviously, such problems can be solved in time proportional to cn. For (1), the 2-dimensional algorithm is based on dynamic programming, while the NP-completeness proof for $d = 3$ is by a reduction from a facility location problem. ∎

3 The Catalog Segmentation Problem

3.1 Dense Instances

The idea of sampling the customer base is pervasive in marketing. We now describe a natural sampling-based approximation scheme for the catalog problem. We give a guarantee on its performance provided $\exists \epsilon > 0$ such that every customer likes at least a fraction ϵ of the items; under a slightly weaker assumption (say, each customer likes $\Omega(d/\log^{O(1)} d)$ of the items) there are instances for which the algorithm does arbitrarily badly. The algorithm runs in time $O(n^{O(k)}/(\delta\epsilon))$ and will, with probability $1 - o(1)$, produce a solution within $1 - \delta$ of the optimal; the failure probability becomes close to 1 as ϵ drops below a constant. Here we outline the algorithm and analysis for $k = 2$.

Denote by $\beta(A, S)$ the value of a catalog A to customer set S, given by the sum over the items in A of the number of customers of S who like each item of A. Any two catalogs A_1 and A_2 induce a partition of all customers into two subsets: those who like more items from A_1 than from A_2, and the rest.

Sample $c \log n$ customers at random. Enumerate all partitions of the sample into 2 subsets. For each subset, find the r items most popular among the customers in that subset. This yields a pair of catalogs for each of the enumerated sample subset pairs. Take the best of all of these enumerations, yielding catalogs B_1 and B_2.

The optimal solution induces a partition of the customers into two subsets S_1 and S_2, and corresponding catalogs A_1 and A_2. If either $\beta(A_1, S_1)$ or $\beta(A_2, S_2)$ is less than $\delta/5$ times the other, we ignore that S_i and focus only on the other subset of customers, whose cardinality (by a sequence of pigeonholing steps) is $\Omega(n)$. Otherwise, both $|S_1|$ and $|S_2|$ are $\Omega(n)$, so each of S_1 and S_2 gets (w.h.p.) a constant fraction of the samples. We will argue (from Lemma 3.1 below) that for $i = 1, 2$ w.h.p. $\beta(B_i, S_i) \geq (1 - \delta)\beta(A_i, S_i)$. The catalogs B_i induce a partition whose value is $\beta(B_i, S_i')$, where S_i' are the customer subsets induced by the B_i; the value of the approximation is thus

$$\begin{aligned}\sum_{i=1}^{2} \beta(B_i, S_i') &\geq \sum_{i=1}^{2} \beta(B_i, S_i) \\ &\geq (1 - \delta) \sum_{i=1}^{2} \beta(A_i, S_i).\end{aligned}$$

It remains to establish the second of the inequalities above. Let R_1 denote the samples intersecting S_1, and R_2 the samples intersecting S_2. When we enumerate all 2-way partitions of the random sample, we will in particular reach a stage when we look at R_1 and R_2 and solve the 1-catalog problem for each. The crucial observation is that the samples of R_i are uniformly distributed on S_i, for $i = 1, 2$. Note that we're not making this argument for arbitrary sets S_i; only for S_1 and S_2 fixed in advance of the sampling.

Lemma 3.1 *Let $a_1, \ldots, a_r$ be the items in the optimal catalog A_i, listed in order of decreasing number of customers of S_i liking the item (henceforth "degree"). Let $a_1, \ldots, a_{r'}$ be the items of A_i with degree $\geq (\delta/5)\beta(A_i, S_i)$ for $r' \leq r$. Let $b_1, \ldots, b_{r'}$ be the r' most popular items in the sample R_i, listed again in order of decreasing degree. With probability $1 - n^{-\Omega(1)}$, for all $1 \leq j \leq r'$,*

$$\mathit{degree}(b_j) \geq (1 - \delta/2)\,\mathit{degree}(a_j).$$

The proof of the lemma is based on showing that for any item x with degree less than $(1 - \delta)\text{degree}(a_j)$, the probability that it beats out a_j in the sample is small. Now, note that the lemma implies that with high probability, $\sum_{i=1}^{2} \beta(B_i, S_i) \geq (1 - \delta) \sum_{i=1}^{2} \beta(A_i, S_i)$.

Theorem 3.2 *On dense instances of the catalog segmentation problem the sampling algorithm will, with probability $1 - o(1)$, yield k catalogs of total value at least $(1 - \delta)$ times the optimal, in time $O(n^{O(k)}/(\epsilon\delta))$.*

3.2 Variable Segmentation

We now consider the catalog segmentation problem in the *variable* case, when the number of catalogs is not set in advance, but there is a fixed cost for each catalog that is produced. For a set S of catalogs, let $g(S)$ denote the utility of producing precisely the set S of catalogs; we are thus seeking to maximize the function $g'(S) = g(S) - \gamma|S|$, for a fixed cost γ.

A very interesting point that arises immediately is the following: when $r = 1$, so that each catalog can contain a single item, $g'(S)$ is simply the cardinality of a union of sets minus a fixed cost for each set. For larger values of r, it is easy to show that g is *monotone* — $g(S) \leq g(T)$ when $S \subseteq T$ — and *submodular* — $g(S) + g(T) \geq g(S \cap T) + g(S \cup T)$.

Thus, we are seeking to maximize a monotone submodular function minus a fixed-cost function; we call such a function a *profit function*. The maximization of a profit function is a basic NP-complete problem whose approximability appears not to be understood at all; the performance of greedy algorithms for profit functions was raised by Berman, Hodgson, and Krass as an open question [4].

In this section, we provide an analysis of the following greedy algorithm for profit functions arising from arbitrary monotone submodular functions.

GREEDY ALGORITHM FOR PROFIT FUNCTIONS: At all times, maintain a candidate solution S. If there is an element $x \notin S$ for which the *marginal gain* $g(S \cup \{x\}) - g(S)$ is at least γ, then add to S an element of maximum marginal gain. Otherwise, terminate and return S.

By re-scaling g, we can assume without loss of generality that $\gamma = 1$ henceforth.

There is no absolute constant c such that the greedy algorithm provides a c-approximation for all profit functions. However, we show that a natural parametrization for analyzing the greedy algorithm turns out to be the quantity μ, defined to be the profit-to-cost ratio of a minimum-size optimal solution T:

$$\mu = \frac{g'(T)}{|T|}.$$

It turns out that the greedy algorithm achieves a constant approximation ratio when μ is constant — i.e., when g is such that a fixed percentage of profit can be made.

First, we can construct an example in which μ sets a natural limit on the performance of the greedy algorithm.

Theorem 3.3 *The greedy algorithm does not achieve a performance guarantee better than* $\frac{\mu}{1+\mu}$.

However, when μ is a constant, the greedy algorithm achieves a constant performance guarantee.

Theorem 3.4 *The greedy algorithm achieves a performance guarantee of at least*

$$\Omega\left(\frac{(\sqrt{1+\mu}-1)^2}{1+\mu}\right).$$

This is $\Theta\left(\mu^2\right)$ *when* $\mu \leq 1$ *and* $\Theta\left(\frac{\mu}{1+\mu}\right)$ *when* $\mu \geq 1$.

The proof of this theorem is somewhat lengthy; for reasons of space we can only provide a brief sketch. When T, the minimum optimal set of catalogs, is large relative to S, then it contains many elements that the greedy algorithm chose not to add to S; this, together with the submodular property, puts an upper bound on $g'(T)$. When T is not much larger than S, we consider two cases. If relatively many of the elements chosen by the greedy algorithm had a marginal gain significantly greater than 1 when they were added to S, then $g'(S)$ is sufficiently large that the bound follows easily. If relatively few of the elements had large marginal gain, then an argument similar to one of Cornuejols, Fisher, and Nemhauser [10] shows that $g'(S)$ and $g'(T)$ must be close.

As a direct corollary of Theorem 3.4, we obtain efficient approximation algorithms for the variable catalog segmentation problem when r is fixed — each step of the greedy algorithm can be implemented by evaluating the effect of producing each possible catalog. For general r, it appears that the greedy algorithm cannot be implemented efficiently. Specifically, given a set S of catalogs, and a cost γ, it is NP-complete to decide whether there is a catalog x for which $g(S \cup \{x\}) - g(S) \geq \gamma$.

It is possible to extend the analysis in Theorem 3.4 to cover the case in which each step of the greedy algorithm is only implemented in an *approximate* sense; we omit the details here. An interesting open question is whether there is an efficient implementation of a suitably strong approximate step of the greedy algorithm, in the case of catalog segmentation; there is some evidence that this will be difficult to obtain.

4 The Hypercube Segmentation Problem

In the HYPERCUBE SEGMENTATION PROBLEM we are given a set S of n customers, each a vertex of the d-cube. We seek k segments $S_1, \ldots, S_k$ and a policy P_i for each i so as to maximize $\sum_{i=1}^{k} \sum_{c \in S_i} P_i \odot c$, where P_i is a vertex of the d-cube and $\odot$ is the Hamming "overlap" operator between two vertices of the d-cube, defined to be the number of positions they have in common. Note that there is a trivial policy P that, without segmentation, yields a benefit of at least 50% of the optimum: pick the majority bit in each of the d coordinates. We give several algorithms that improve on this 50% figure. We first show that if we restrict ourselves to policies that are customers, the loss is modest. Consider any set T of m vertices of the d-cube, $v_1, \ldots, v_m$.

Lemma 4.1 *Let* P *denote the optimal policy for* T. *Then there exists a customer* v_i *such that*

$$\sum_{j=1}^{m} v_i \odot v_j \geq (2\sqrt{2}-2) \sum_{j=1}^{m} P \odot v_j. \tag{1}$$

The bound $(2\sqrt{2}-2) \approx .828$ of Lemma 4.1 cannot be improved significantly; there are examples in which no customer gets better than $5/6 \approx .833$ of the benefit of the optimal policy. Let $T_1, \ldots, T_k$ denote the segments of the optimal segmentation; by Lemma 4.1, $\forall i$ there exists a customer $t_i \in T_i$ such that using t_i as the policy for T_i yields at least $(2\sqrt{2}-2)$ of the contribution of T_i to the total benefit of the optimal segmentation. By enumerating all k-subsets of S, we can find the policies $t_1, \ldots, t_k$ in time $O(kn^{k+1})$, for a $(2\sqrt{2}-2)$-approximation to the optimal segmentation (note that the segmentation induced by $t_1, \ldots, t_k$ need not be $T_1, \ldots, T_k$).

The above is feasible for small k; for larger k, we have two additional approximation algorithms, whose details we omit. The idea is to use random sampling together with an analysis of the "occupancy problems" that result. Overall we can prove:

Theorem 4.2 *The* HYPERCUBE SEGMENTATION PROBLEM *can be approximated as follows:*

1. *Within* .828 *by an* $O(kn^{k+1})$ *deterministic algorithm.*

2. *Within* 0.7*, with high probability, by an* $O(c^k n)$ *randomized algorithm, where* c *is a constant independent of* k *and* n*.*

3. *Within* $.828 - .328e^{-\ell/2k}$*, with high probability, by an* $O(n\ell)$ *randomized algorithm that will approximate the optimum* k*-segmentation by* ℓ *policies. (This ratio is roughly* 0.63 *for* $\ell = k$*,* .7 *when* $\ell = 2k$*, and is asymptotic to* .828*.)*

5 Weakly Submodular Functions

In this section, we consider a general framework for analyzing fixed segmentation problems. We will find that the definition of a submodular function is too restrictive to cover the objective functions for general segmentation problems; thus we introduce the notion of a *weakly submodular function* and analyze a natural greedy algorithm in terms of such functions.

As before, let $\mathcal{D}$ be a set of possible decisions, and $\mathcal{C}$ a set of customers with associated functions f_i. If $S \subseteq \mathcal{D}$, we define $\sigma(S)$ to be $\sum_{i=1}^{n} \max_{x \in S} f_i(x)$. We call a function σ arising in this way a *segmentation function.*

Now, the fixed segmentation problem for $(\mathcal{D}, f)$ can be phrased as follows: for the associated segmentation function σ, find a set S of size k that (approximately) maximizes $\sigma(S)$. This phrasing of the problem resembles the maximization problem for monotone submodular functions, studied by Cornuejols, Fisher, and Nemhauser [10] and Nemhauser, Wolsey, and Fisher [19]; we drew a different but related connection to this work in Section 3.2.

However, it turns out that segmentation functions need not be submodular. As an example, let $\mathcal{D}$ be the L_2 unit ball, let v be a unit vector in $\mathbf{R}^d$, and let $\mathcal{C}$ consist of two customers, so that $f_1(x) = v \cdot x$ and $f_2(x) = -v \cdot x$. Then

$$\sigma(\{v\}) + \sigma(\{-v\}) = 0 + 0 < \sigma(\phi) + \sigma(\{v, -v\}) = 0 + 2,$$

which violates the submodular property.

We show here that segmentation functions can be meaningfully studied in terms of a more general notion, which we call the *weak submodularity* property. We say that a set function g is *weakly submodular* if $g(S) + g(T) \geq g(S \cap T) + g(S \cup T)$ for all pairs of sets S, T that have non-empty intersection.

First we show

Lemma 5.1 *Every segmentation function* σ *is weakly submodular.*

Another issue to deal with is that segmentation functions are not necessarily monotone — specifically, on singleton sets. We thus say that a set function g satisfying $g(\phi) = 0$ is *non-degenerate* if there exists a singleton set $S = \{x\}$ for which $g(S) \geq 0$.

Using these properties, we analyze the following basic greedy algorithm.

GREEDY ALGORITHM FOR SEGMENTATION FUNCTIONS: At all times, maintain a candidate solution S. While $|S| < k$, add an element $x \notin S$ which maximizes the *marginal gain* $g(S \cup \{x\}) - g(S)$.

We can now show

Theorem 5.2 *Let* g *be a non-degenerate segmentation function. Then the greedy algorithm achieves a performance guarantee of* $1 - (1 - 1/k)^{k-1}$*, which converges to* $1 - 1/e$ *from below as* k *increases.*

The proof is based heavily on the weak submodularity property.

If we can only approximate each step of the greedy algorithm to within a factor of $c > 1$, then the analysis in the proof of Theorem 5.2 can be adapted to show a performance guarantee of $1 - (1 - \frac{1}{ck})^{k-1}$, which converges to $1 - 1/e^{1/c}$ from below as k increases.

Although Theorem 5.2 applies to all non-degenerate segmentation functions, we observed in Section 3.2 that the implementation of a single step of the greedy algorithm can sometimes be an NP-complete in its own right; thus, this theorem does not directly yield an *efficient* approximation algorithm. However, Theorem 5.2 does provide an efficient approximation algorithm for segmentation problems arising from optimization over polyhedra that contain the origin and have a polynomial number of vertices. In such a case, the greedy algorithm need only examine vertex solutions, and thus can run in polynomial time; one can also verify that non-degeneracy holds here. A basic example captured by this setting is an arbitrary segmentation problem on the L_1 unit ball, which generalizes the HITTING SET problem.

Theorem 5.3 *There is an efficient* $[1 - (1 - \frac{1}{k})^{k-1}]$*-approximation algorithm for a* k*-segmentation problem in which the underlying objective function is a linear program over a polyhedron that contains the origin and has a polynomial number of vertices.*

6 Further Results

6.1 The k-fold Simplex Method

If $\mathcal{D}$ is a linear programming problem described by a convex polytope, then there is a very natural extension of the simplex algorithm [11, 20] that can be used to attack the segmentation problem corresponding to $\mathcal{D}$. In the description of the algorithm, if $(x_1, \ldots, x_k)$ is a k-tuple of vertices of $\mathcal{D}$, by $v(x_1, \ldots, x_k)$ we denote their total utility, that is,

$$v(x_1, \ldots, x_k) = \sum_{i=1}^{n} \max_{1 \le j \le k} v_i \cdot x_j .$$

```
k-fold simplex method:

let x_1,...,x_k := the optimum vertex of D
under objective v_1 + ··· + v_n

while there is a j ≤ k and a vertex x'_j
of D adjacent to x_j such that
  v(x_1,...,x'_j,...,x_k) > v(x_1,...,x_j,...,x_k)
do
     x_j := x'_j
```

That is, we start with all k solutions coinciding with the optimum single solution, and we keep improving the ensemble of solutions by sliding them one-by-one along edges of the polytope. We halt when a local optimum has been identified. Unfortunately, we cannot prove much about the worst-case performance of this algorithm — beyond the obvious fact that it cannot do worse than $1/k$ times the optimum — and we feel that a non-trivial analysis for some concrete segmentation problem is an interesting open question.

We have performed some preliminary experiments with the k-fold simplex method in the context of the catalog segmentation problem with $k = 2$; these suggest that for certain ranges of the parameters, the algorithm achieves fairly strong performance. In this setting, the method degenerates to the *2-opt* local search heuristic. That is, we maintain two sets of size r, and repeatedly try to improve the total revenue from the two sets by taking some element out of one of the sets, and replacing it with another element. In a number of experiments on randomly-generated instances, 2-opt beat (by between 10 and 40%) the initial solution, in which both catalogs were set to contain the d most popular items. This begs the following open question: can 2-opt do better than 50% of the optimal solution in the worst case?

6.2 Segmentation in a Model of Competition

So far we have considered the problem faced by a single enterprise trying to optimize its utility. But it is natural to try incorporating the notion of segmentation problems into a setting that involves competition among several enterprises; to indicate some of the issues that arise, we consider the classical framework of *two-player games.*

Thus, consider the following type of *segmented matrix game*. There are n customers, each with an $m_1 \times m_2$ *pay-off matrix* M_i. Let C denote the set of matrices. There are two players, P_1 and P_2. P_1 partitions C into k_1 sets, and chooses a row-strategy for each set. We will consider both pure strategies (choosing a single row) and mixed strategies (choosing a distribution over rows). P_2 then does the same, using k_2 column-strategies. We will assume throughout that P_1 is trying minimize the pay-off, while player P_2 is trying is to maximize it.

It is possible to consider models with either one round or several rounds; and in which the players move either sequentially or simultaneously. For the sake of concreteness here, let us consider a one-round game in which P_1 moves first — revealing his move — and P_2 then moves.

First consider the problem faced by P_2. Once P_1 reveals his move (whether it is pure or mixed), each matrix M_i is collapsed to a single row vector v_i (either the selected row or a weighted average of rows). Consider the problem now faced by P_2 in choosing k_2 mixed strategies: she must determine k_2 non-negative vectors each of whose coordinates sum to 1. Thus, we have an LP segmentation problem over the intersection of the hyperplane $x_1 + \ldots x_{m_2} = 1$ with the positive orthant. Note that the k_2 vectors may as well be vertex solutions, i.e., pure strategies. There is a difficulty in analyzing the performance of the greedy algorithm for this particular segmentation problem (c.f. Section 5): the polyhedron defining the feasible region does not contain the origin, and so it is possible that no single solution has non-negative pay-off for P_2. As such, we do not have the *non-degeneracy* condition needed for the $1 - (1 - 1/k)^{k-1}$ bound. Indeed, it can be NP-complete to decide whether there is a set of k_2 pure strategies for P_2 that produce a non-negative pay-off. For consider an instance of the HITTING SET problem, with sets $Z_1, \ldots, Z_n \subseteq U = \{u_1, \ldots, u_{m_2}\}$. Map Z_i to a vector v_i, whose j^{th} entry is 0 if $u_j \in Z_i$, and is -1 otherwise. Then there is a collection of k_2 elements hitting all the sets Z_i if and only if P_2 can achieve a pay-off of 0 (the maximum possible).

Now let us consider the problem faced by P_1, assuming that P_2 is computationally unbounded (while P_1 runs in polynomial time). To make the discussion more combinatorial, let us suppose that each matrix M_i has ± 1 entries; so at the end of the game, each customer is "won," in a discrete way, by either P_1 or P_2. Let us also assume that both players are restricted to pure strategies.

The SET UNION problem asks, given a collection of sets $S_1, \ldots, S_t$, to form a sub-collection of k of these sets whose union is as large as possible. In these terms, P_1 is faced with the following problem. First, he chooses k_1 indices between 1 and m_1. He then maps each customer i to one of the k_1 rows of the matrix M_i corresponding to the chosen indices. Suppose that i is mapped to v_i, and let S_j denote the set of i for which v_i has a $+1$ in column j. P_2 can then choose k_2 column indices; her pay-off is determined by the cardinality of the union of the corresponding sets S_j. Thus, P_1 is attempting to create an instance of the SET UNION problem whose optimum is as small as possible.

Thus we are dealing with an optimization problem in which the objective is to create problem instances with poor-quality optima. Perhaps the cleanest version of the above situation is the one in which $k_1 = n$; for then P_1 can construct a SET UNION problem by picking one set from the options provided by each customer, rather than having to deal with the additional constraint that they must come from a small number (k_1) of distinct rows. Thus, if $k_1 = n$ and $k_2 = 1$, we have the following problem: P_1 must choose one ± 1-vector from among the options provided by each customer in such a way that the sum of the resulting vectors has a maximum coordinate that is minimum. This is NP-complete, by a reduction from e.g. the INDEPENDENT SET problem.

A number of interesting issues arise if we drop the assumption that P_2 is computationally unbounded, assuming instead, for example, a polynomial-time limitation. Intuitively, one could imagine that P_1 need no longer only produce problem instances that have poor-quality optima, but could alternatively try to produce instances which are computationally difficult (even if the optimum is good for P_2). We intend to explore these issues in further research.

6.3 Data Mining as Sensitivity Analysis

There are a number of problems involving the aggregation of customer data that do not appear to fall within the framework of segmentation problems. We consider one here, based on correlations among consumer preferences.

A motivating example: Beer and Diapers.[4] Suppose that a retailer stocks two products in quantities x_1 and x_2; the amounts (x_1, x_2) to be stocked are the only decision variables, bounded above by capacity: $x_1 + x_2 \leq c$. The profit margins in the two products are m_1 and m_2. We have a table with 0-1 values $(y_{1,i}, y_{i,2})$ for each customer $i \in \mathcal{C}$, indicating whether the customer will buy a unit of each of the products. That is, in this toy example demand is known deterministically.

In the first scenario, customers arrive in random order, and buy whatever part of their desired basket is available. The revenue of the enterprise is in this case a function of x_1, x_2, m_1, m_2, and the aggregate demands $Y_1 = \sum_{i \in \mathcal{C}} y_{1,i}$ and $Y_2 = \sum_{i \in \mathcal{C}} \cdot y_{i,2}$. Aggregation would not distort the optimal decision, and data mining is moot.

But suppose that the customers arrive in random order, and buy their desired basket in an all-or-nothing fashion; if not all items are available, the customer buys nothing. The expected profit for the enterprise from customer i is now of the form $B_1 \cdot y_{1,i} + B_2 \cdot y_{i,2} + B_3 \cdot y_{1,i} \cdot y_{i,2}$. Because of the nonlinear term, associations between the $y_{1,i}$ and the $y_{i,2}$ columns are now important, and the optimum decision by the enterprise depends critically on them. Aggregation will no longer do the trick, and data mining is desirable, even necessary.

We propose that *associations and correlations between attributes in a table are interesting if they correspond to nonlinear terms in the objective function of the enterprise*, that is, whenever $\frac{\partial^2 g}{\partial y_i \partial y_j} \neq 0$ for the cost function g and some attributes y_i, y_j. We now turn to a formulation that makes this notion precise.

A concrete formulation. The field of *sensitivity analysis* in optimization seeks to develop principles and methodologies for determining how the optimum decision changes as the data change. In this section we give an extensive example suggesting that many common data mining activities can be fruitfully analyzed within this framework.

To fix ideas, we shall consider the case in which the optimization problem facing the enterprise is a *linear program* [11, 20], that is, we have an $m \times n$ matrix A (the *constraint matrix*, $m < n$), an m-vector b (the *resource bounds*), and an n-row vector c (the *objective function coefficients*), and we seek to

$$\max_{Ax=b, x \geq 0} c \cdot x. \tag{1}$$

The columns of A —the components of x— are called *activities*, and the rows of A are called *constraints*. Extensions to non-linear inequalities and objectives are possible, with the Kuhn-Tucker conditions [3] replacing the sensitivity analysis below.

We assume, as postulated in the introduction, that the entries of A and b are fixed and given (they represent the endogenous constraints of the enterprise), while the coefficients of c depend in complex ways on a relation, which we denote by $\mathcal{C}$ (we make no distinction between the relation $\mathcal{C}$, and its set of rows, called *customers*). The ith tuple of $\mathcal{C}$ —the ith customer— is denoted y_i, and we assume that c_j is just $\sum_{i \in \mathcal{C}} f_j(y_i)$, where f_j is a function mapping the product of the domains of $\mathcal{C}$ to the reals. We noted in the introduction that the desirability of data mining depends on whether the functions f_j are "non-linear", that is, on whether the derivatives $\frac{\partial^2 f_j}{\partial y^k \partial y^\ell} \neq 0$ for some k and ℓ.

Assume for the sake of concreteness that all attributes of the relation $\mathcal{C}$ are real numbers in the range $[0, 1]$, and that f_j depends on two attributes, call them k_j and ℓ_j. We also assume that we have an estimate $D_j \geq 0$ on the absolute value of the derivative $\frac{\partial^2 f_j}{\partial y^{k_j} \partial y^{\ell_j}}$. $D_j > 0$ means that f_j is nonlinear. We investigate under what circumstances it is worthwhile to measure the correlations of the pair of attributes corresponding to the coefficient c_j —that is to say, to data mine the two attributes related to the jth activity. Without data mining, the coeffi-

[4] The correlation between the amount of beer and the amount of diapers bought by consumers is one of the delightful nuggets of data mining lore.

cient c_j will be approximated by the function f_j of the aggregate values of the attributes k_j and ℓ_j.

Suppose that we have solved the linear program (1) of the enterprise, based on the aggregate estimation of the c_j's, and let $B^{-1}A = X$ be the simplex tableau at optimality, and $\overline{c} = c - c_B B^{-1} A \geq 0$ its zeroth row, where B is the optimum basis. The theory of sensitivity analysis of linear programming [11, 20] then implies the following qualitative notion of *interestingness* of activity j (intuitively, the degree to which it is worth mining the correlations of the data related to activity j; we omit from this version the precise formal definition):

An activity j is interesting if the function f_j has a highly non-linear cross-term, and either $\overline{c_j}$ is small, or the jth row of the tableau has large positive coefficients at columns with small $\overline{c_i}$'s.

As the above analysis suggests, our point of view, combined with classical linear programming sensitivity analysis, may lead to a quantitative theory for determining when data mining can affect decisions, ultimately to *a theory for predicting the value of data mining operations.*

7 Conclusions and Open Problems

The class of *segmentation problems* arises from the aspects of data mining that can be viewed as "clustering with an economic objective function." Our hope in introducing this model is to offer a particular algorithmic perspective on the *value* of "mined data," in terms of this underlying objective function, and to indicate the surprisingly wide range of concrete optimization problems that arise from this point of view.

There are many open questions arising from this work, and we suggest some that seem to be among the most interesting. First, obtaining good approximation algorithms for the general CATALOG SEGMENTATION PROBLEM appears to quite difficult; for example, in the general case, we do not know how to improve on the trivial $\frac{1}{2}$-approximation when $k = 2$. We also do not know very much about the approximability of segmentation problems arising from traditional graph optimization problems such as the MST and TSP.

One can also formulate versions of the basic segmentation problem different from the two (fixed and variable) that were studied here. For example, suppose each customer has a function f_i and a *threshold* s_i; the customer is *satisfied* by a policy x if $f_i(x) \geq s_i$. We may then be required to implement k policies so as to satisfy as many customers as possible. Again, many optimization problems can be studied in this setting.

We have suggested two general-purpose approaches for approximating segmentation problems: the greedy algorithm of Section 5, and the k-fold simplex method. In the former case, it is unfortunate that the *implementation* of a single step of the greedy algorithm can sometimes be an NP-complete problem; we are interested in determining the range of settings in which an *approximate greedy algorithm* can be usefully applied. In the latter case, we feel it would be interesting — both in the context of segmentation problems and as a natural problem for the analysis of local search heuristics — to find a setting in which one could prove non-trivial approximation bounds for the k-fold simplex method.

Finally, the models proposed in Section 6.2 and 6.3 suggest a number of directions to explore.

References

[1] R. Agrawal, T. Imielinski, A. Swami. "Mining association rules between sets of items in a large database," *1993 SIGMOD*, pp. 207–216, 1993.

[2] S. Arora, D. Karger, M. Karpinski. "Polynomial-time approximation schemes for dense instances of NP-hard problems," *Proc. ACM STOC*, 1995.

[3] M. Avriel. *Nonlinear Programming: Analysis and Methods.* Prentice-Hall, 1976.

[4] O. Berman, M.J. Hodgson, D. Krass. "Flow-interception problems," in *Facility Location: A Survey of Applications and Methods*, Z. Drezner, Ed., Springer 1995.

[5] M. Bern, D. Eppstein. "Approximation algorithms for geometric problems," in *Approximation Algorithms for NP-Hard Problems*, D. Hochbaum, Ed., PWS Publishing, 1996.

[6] M. J. Berry, G. Linoff. Data Mining Techniques. John-Wiley, 1997.

[7] S. Brin, R. Motwani, J.D. Ullman, S. Tsur. "Dynamic itemset counting and implication rules for market basket data". *Proc. ACM SIGMOD*, 1997.

[8] M. S. Chen, J. Han, P. S. Yu. "Data mining: An overview from a database perspective." *IEEE Trans. on Knowledge and Data Eng., 8*, 6, pp. 866–884, 1996.

[9] J.M. Coggins. "Dissimilarity measures for clustering strings," in *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*, D. Sankoff and J.S.B. Kruskal, Eds., Addison-Wesley, 1983.

[10] G. Cornuejols, M. Fisher, G. Nemhauser. "Location of Bank Accounts to Optimize Float," *Management Science*, 23(1977), pp. 789–810.

[11] G. B. Dantzig. *Linear programming and Extensions.* Princeton Univ. Press, 1963.

[12] T. Feder, D. Greene. "Optimal algorithms for approximate clustering," *Proc. ACM STOC*, 1988.

[13] T. Gonzalez. "Clustering to minimize the maximum inter-cluster distance, *Theoretical Computer Science*, 38(1985), pp. 293–306.

[14] D. Gunopoulos, R. Khardon, H. Mannila, H. Toivonen. "Data mining, hypergraph transversals, and machine learning". *Proc. 1997 PODS,* pp. 209–217, 1997.

[15] A. K. Jain, R. C. Dubes. *Algorithms for Clustering Data*, Prentice-Hall, 1981.

[16] M. Kearns, Y. Mansour, A. Ng. "An information-theoretic analysis of hard and soft assignment methods for clustering," *Proc. 13th Conference on Uncertainty in Artificial Intelligence*, 1997.

[17] B. Liu and W. Hsu. "Post-analysis of learned rules," *Proc. AAAI*, pp. 828–834, 1996.

[18] B.M. Masand and G. Piatetsky-Shapiro. "A comparison of approaches for maximizing business payoff of prediction models". *Proc. Knowledge Discovery and Data Mining,* 195–201, 1996.

[19] G. Nemhauser, L. Wolsey, M. Fisher. "An analysis of the approximations for maximizing submodular set functions," *Mathematical Programming*, 14(1978), pp. 265–294.

[20] C. H. Papadimitriou, K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity* (second edition). Dover, 1997.

[21] G. Piatetsky-Schapiro, C. J. Matheus. "The interestingness of deviations," *Proc. Knowledge Discovery and Data Mining,*, pp. 25–36, 1994.

[22] D. Shmoys, É. Tardos, K. Aardal. "Approximation algorithms for facility location problems," *Proc. ACM STOC*, 1997.

[23] A. Silberschatz and A. Tuzhilin. "What makes patterns interesting in knowledge discovery systems ," *IEEE Trans. on Knowledge and Data Eng., 8*, 6, 1996.

[24] P. Smyth, R. M. Goodman. "Rule induction using information theory," *Proc. Knowledge Discovery and Data Mining*, 1991.