Segmentation Problems

JON KLEINBERG
Cornell University
and
CHRISTOS PAPADIMITRIOU
UC Berkeley
and
PRABHAKAR RAGHAVAN
Verity

We study a novel genre of optimization problems, which we call *segmentation problems*, motivated in part by certain aspects of clustering and data mining. 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. We focus on two natural and interesting (but MAXSNP-complete) problems in this class, the HYPERCUBE SEGMENTATION PROBLEM and the CATALOG SEGMENTATION PROBLEM, and present approximation algorithms for them. We also present a general greedy scheme, which can be specialized to approximate any segmentation problem.

Categories and Subject Descriptors: F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Non-numerical Al gorithms and Problems; H.2.8 [**Database Management**]: Database Applications – Data Mining

General Terms: Algorithms, Theory

Additional Key Words and Phrases: Clustering, data mining, approximation algorithms, market segmentation

1. INTRODUCTION

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

This work has been supported in part by grants from the NSF and from the David and Lucile Packard Foundation. A preliminary version of this work appeared at the 30th ACM Symposium on Theory of Computing, 1998.

of U, each of size r, so that

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

is maximized.

This is only one in a novel and interesting class of combinatorial optimization problems, which we call *segmentation problems*, introduced and studied in this paper. Like almost all such problems, it is NP-hard, even in the unit-weight case formulated above. One can define a segmentation problem (and in fact one of several variants) for every classical optimization problem. Segmentation problems are intended to capture certain aspects of the economic basis for *data mining* [Kleinberg et al. 1998] and *clustering;* we explain this connection next.

The Value of Data Mining

Data mining is the application of statistical and machine-learning techniques for *extracting interesting patterns from raw data*. The problem of what "interesting" means in this context has been an important issue in the data mining literature, but there has been very little work on formalizing the notion. (See, e.g., [Agrawal et al. 1993; Masan and Piatetsky-Shapiro 1996; Piatetsky-Shapiro and Matheus 1994; Smyth and Goodman 1991; Silberschatz and Tuzhilin 1996] for various perspectives on the problem). Most research in data mining deals with the efficient discovery of patterns for subsequent human evaluation of the degree to which they are "interesting," and not on techniques for automatically evaluating mined patterns, or for automatically focusing on "interesting" patterns. We proposed in [Kleinberg et al. 1998] a rigorous and algorithmic framework for such evaluation *based on the pattern's utility in decision-making*. The framework formulated in [Kleinberg et al. 1998] suggests a number of interesting computational issues, related to sensitivity analysis and clustering; in the present work we study algorithms for one class of such optimization problems — the *segmentation problems*.

Segmentation problems address the problem of the *degree of aggregation* in the data that an enterprise uses for decision-making. Any enterprise faces an optimization problem

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

where $\mathcal{D}$ is the domain of feasible decisions (production plans, marketing strategies, etc.), and $f(x)$ is the utility of decision $x \in \mathcal{D}$. The function f is generally, to a first approximation, a large superposition of functions $f_i(x)$, $i \in \mathcal{C}$, reflecting the enterprise's interaction with a set $\mathcal{C}$ of agents (customers, suppliers, employees, and other external factors affecting the utility of the enterprise; we will refer to them as "customers").

As a concrete example: A company has information about a set $\mathcal{C}$ of customers, and a choice of various marketing strategies $x \in \mathcal{D}$. Any given marketing strategy 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 $\sum_{i \in \mathcal{C}} f_i(x)$.[1]

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

[1]This sum ignores inter-dependencies among the customers $i \in C$; but for now we will focus on this first approximation for concreteness. The issue of non-linearities in the data, in a somewhat different context, is addressed in [Kleinberg et al. 1998].

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, mail-order companies produce several hundred different catalogs each year, targeting one or more at each of the customers on their mailing list. A telephone company may divide its customers into two segments: residence and business customers; they offer different terms and prices to the two. 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. (In Section 5 we formulate and solve the PRICE SEGMENTATION problem very much along the lines of the telephone company example.)

Segmentation problems also relate to *clustering*, an important, classical, and challenging algorithmic problem area [Jain and Dubes 1981], of interest in data mining —which it predates. Suppose that a large set of points in a high-dimensional 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 [Bern and Eppstein 1996; Coggins 1983; Feder and Greene 1988; Gonzalez 1985; Kearns et al. 1997]) but very little problem-independent guidance on how to select the most appropriate one. But suppose that we are told (to pick a toy example here) that the dimensions of the space correspond to the edges of a fixed graph, and the points in the space represent the weights of the edges as perceived by different people interested in minimum spanning trees of the graph. Given this additional information, the natural approach to clustering the points would to solve the MST SEGMENTATION problem, and thereby partition the points according to the spanning tree each group prefers. (See Theorem 5.1 in Section 5 for another concrete example of this sort).

One of the main advantages of clustering has always been that it enables flexible decision-making at the cluster level. Segmentation problems use explicitly the underlying decision-making problem as the clustering criterion.

The General Segmentation Problem

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 $f_i : \mathcal{D} \to \mathbf{R}$. 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 $\sum_{i=1}^{n} \max_{1 \le j \le k} f_i(x_j)$ is maximized.

A strictly analogous definition can be made for minimization problems. We thus have the MST SEGMENTATION PROBLEM, the TSP SEGMENTATION PROBLEM, and so on. In this paper we focus on certain natural segmentation problems that capture the marketing motivation. We can define another version:

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. However, the partition version is not totally devoid of algorithmic advantages: Some of our approximation algorithms involve solving the problem with only a logarithmic sample of customers; in this context, the exponential naive algorithm for the partitioning problem becomes attractive.

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, the problems we consider here 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 approximation algorithms for the discrete facility location problem and its variants, on the other hand (see e.g. [Shmoys et al. 1997]), the facilities must typically be sited at demand locations, yielding, immediately, a relatively small space of possible decisions.

Complexity

In [Kleinberg et al. 1998], we prove certain negative complexity results about segmentation problems. In particular, we show that many natural versions are NP-complete, in several cases even when the un-segmented version of the underlying optimization problem is trivially solvable.

THEOREM 1.1. (See [Kleinberg et al. 1998].) *The segmentation problems corresponding to the following feasible sets $\mathcal{D}$ are MAXSNP-complete (with linear objective functions $\{f_i\}$ in each case): (1) The d-dimensional unit ball, even with $k = 2$; (2) the d-dimensional unit L_1 ball; (3) the d-dimensional hypercube, even with $k = 2$; (4) the r-slice of the d-dimensional hypercube (the* CATALOG SEGMENTATION PROBLEM*), even with $k = 2$; (5) the set of all spanning trees of a graph G, even with $k = 2$.*

THEOREM 1.2. (See [Kleinberg et al. 1998].) *Segmentation problems (2–5) above can be solved in time polynomial in n 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 MAXSNP-complete in three dimensions.*

Overview of the Results of this Paper

We focus on approximation algorithms for two concrete problems: the CATALOG SEGMENTATION PROBLEM introduced at the outset, and the HYPERCUBE SEGMENTATION PROBLEM. In this latter problem, both customers and policies are vertices of a d-cube, and the utility of implementing a given policy with respect to a given customer is equal to their *Hamming overlap* — the number of bits on which they agree. A number of the techniques we develop for these problems apply, in fact, in much greater generality.

In Section 2.1 we give a natural sampling-based approximation scheme for the CATALOG SEGMENTATION PROBLEM. Somewhat surprisingly, the algorithm can only be shown to work under a fairly strong *density*

assumption on instances (akin to the work of Arora, Karger and Karpinski [1995]). Even in this case the analysis is non-trivial, and under slightly weaker assumptions on instances, the algorithm is known not to work.

In Section 2.2, we analyze a natural greedy algorithm for the VARIABLE CATALOG SEGMENTATION PROBLEM, in which there is a fixed cost for each catalog produced. Our analysis is carried out at the more general level of an arbitrary *monotone submodular function* minus a fixed-cost function; it thus forms a natural parallel to results of Nemhauser, Wolsey, and Fisher [Cornuejols et al. 1977; Nemhauser et al. 1978] on the maximization of monotone submodular functions, and addresses an open question raised by Berman, Hodgson, and Krass [1995].

In Section 3 we consider the HYPERCUBE SEGMENTATION PROBLEM, defined above. We wish to approximate the optimal segmentation into k sets. We give three approximation algorithms: (i) a deterministic algorithm running in time $O(kn^{k+1})$, yielding a segmentation that is at least 0.82 times the optimal; (ii) a randomized algorithm running in time nc^k, yielding an approxation ratio of $.82 - \epsilon$, where $\epsilon > 0$ is arbitrary and c depends on the value of ϵ; (iii) a randomized algorithm that uses ℓ segments, running in time $O(n\ell)$, achieving an approximation ratio that approaches 0.82 as ℓ becomes large; for $\ell = k$ it yields a 0.63-approximation.[2]

Finally, in Section 4, we present a general framework, based on a greedy algorithm, for approximating any segmentation problem. The analysis is again motivated by connections to the maximization of monotone submodular functions; however, the global objective functions arising from general segmentation problems need not be either submodular or monotone. We therefore introduce the notion of a *meta-submodular function*, which captures the key properties of the objective functions in segmentation problems; and we analyze the greedy algorithm for such functions. We note that there are natural cases in which the greedy algorithm may not be useful in developing efficient algorithms, for the implementation of a single step of the greedy algorithm can sometimes be an NP-complete problem in its own right; see the discussion at the end of Section 2.2.

2. THE CATALOG SEGMENTATION PROBLEM

2.1 Dense Instances

The idea of sampling a 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 there exists an $\epsilon > 0$ such that every customer likes at least a fraction ϵ of the items; under a slightly weaker assumption (say, each customer likes $\Omega(d/\log^{\Omega(1)} d)$ of the d items) there are instances for which the algorithm does arbitrarily badly. Thus, sampling will (on such "dense" instances) essentially capture the benefit of the optimal segmentation.

Given an arbitrary parameter $\delta > 0$, the algorithm runs in time $O((n+d)^{O(k \log k)}/(\delta\epsilon))$ 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 give the algorithm and analysis for $k = 2$.

Denote by $\text{Benefit}(A, S)$ the value of a catalog A to customer set S, given by the sum over each item in A of the number of customers of S who like this item. In other words, if we let $L(x)$ denote the set of all customers who like item x, then we have

$$\text{Benefit}(A, S) = \sum_{x \in A} |S \cap L(x)|.$$

[2]Following the preliminary conference version of this work, Alon and Sudakov [1999] developed algorithms improving some of these bounds; we discuss this work in Section 5.

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 $c\log(n+d)$ customers at random; the constant c (and implicitly the base of the logarithm) depend on the probability estimates below. 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 these $(n+d)^c$ enumerations, yielding catalogs B_1 and B_2.

By the density assumption, the value of the optimal segmentation is at least ϵrn. 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 Benefit(A_1, S_1) or Benefit(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 (with high probability) a constant fraction of the samples. We will argue (from Lemma 2.1 below) that for $i = 1, 2$ Benefit$(B_i, S_i) \geq (1-\delta)$Benefit$(A_i, S_i)$ with high probability. The catalogs B_i induce a partition whose value is Benefit(B_i, S_i'), where S_i' are the customer subsets induced by the B_i; the value of the approximation is thus

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

It remains to establish the second of the inequalities above. Let R_1 denote the samples in S_1, and R_2 the samples in S_2. When we enumerate all 2-way partitions of the random sample, we will in particular consider the partition R_1, R_2 and solve the 1-catalog problems for R_1 and R_2. The crucial observation is that the samples of R_i are uniformly distributed on S_i, for $i = 1, 2$. Note that we do not make this argument for arbitrary sets S_i; only for S_1 and S_2 fixed in advance of the sampling.

The following lemma applies for $i = 1, 2$.

LEMMA 2.1. *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 $\geq (\delta/5r)$Benefit(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 $1 - (n+d)^{-\Omega(1)}$, for all $1 \leq j \leq r'$,*

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

The proof of the lemma is based on showing that for any item x with degree less than $(1-\delta/2)$degree(a_j) $(1 \leq j \leq r')$, the probability that x beats out a_j in the sample (by being preferable to more customers in the sample) is small. Further, the contribution of items $a_{r+1}, \ldots, a_r$ to Benefit(A_i, S_i) is at most $r \times (\delta/5r)$Benefit$(A_i, S_i) \leq (\delta/5)$Benefit$(A_i, S_i)$. Now, note that the lemma implies that with high probability, $\sum_{i=1}^{2}$ Benefit$(B_i, S_i) \geq (1-\delta/2-\delta/5)\sum_{i=1}^{2}$ Benefit(A_i, S_i).

Proof of Lemma 2.1: For any item x with degree less than $(1-\delta/2)$degree(a_j), we will show that the probability that it beats out a_j in the net preference of the sample is small. The sample will prefer x to a_j only if it picks more customers who like x than who like a_j. A sample customer can like only a_j, only x, neither or both. Thus the number of samples preferring a_j to x is binomially distributed with mean $\geq (c'\log(n+d))/(2-\delta/2)$ for a constant c'. Since R_i is $\Omega(\log n)$, and these samples are uniformly distributed in S_i, any single challenger x upsets a_j with probability at most $(n+d)^{-c''}$, where c'' is proportional to c'. Since there are at most d possible challengers x and $r < n$ possible a_j, the overall probability of failure is at

most $(n+d)^{-c''+1}$, which can be made small by choosing a suitably large value of c (and thus c' and c''). ■

THEOREM 2.2. *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* $O((n+d)^{O(k \log k)}/(\epsilon\delta))$.

2.2 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.

The following interesting point arises immediately: 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 [Berman et al. 1995].

In this section, we provide an analysis of the following greedy algorithm for profit functions arising from arbitrary monotone submodular functions. We will assume without loss of generality that for every profit function g that we deal with, we have $g(\phi) = 0$.

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 x to S. Otherwise, terminate and return S.

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

We seek bounds r such that the profit obtained by the greedy algorithm is at least a factor of r times the profit of an optimal solution; we refer to such an r as a *performance guarantee* for the greedy algorithm. As we will see, 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 is 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 construct an example in which μ sets a natural limit on the performance of the greedy algorithm.

THEOREM 2.3. *The greedy algorithm does not achieve a performance guarantee better than* $\mu/(1+\mu)$.

PROOF. Let X be a finite set, and Y_i, for $i \in U$, a collection of subsets of X. Fix a positive constant $\epsilon < 1$ so that $1/\epsilon$ is an integer, and set $c = 1/\epsilon^2$. We define

$$g(S) = \frac{1}{c}\left|\bigcup_{i \in S} Y_i\right|.$$

Now, the sets in $\{Y_i\}$ are defined as follows. $Y_1, \ldots, Y_k$ are all disjoint and have cardinality $(1+\epsilon)c$. Partition $\cup_{i=1}^{k} Y_i$ into disjoint sets $Z_1, \ldots, Z_{\epsilon^{-1}(1+\epsilon)k}$ of size ϵc each. Now define Y_{k+j}, for $j = 1, \ldots, \epsilon^{-1}(1+\epsilon)k$ to be

a set of size $(1+\epsilon)c$ containing Z_j and otherwise disjoint from all other Y_i. Then the greedy algorithm will pick the sets $Y_1, \ldots, Y_k$, for a profit of ϵk; but the (smallest) optimum solution is to pick all other sets, for a profit of $(1+\epsilon)k$. In this instance, the profit-to-cost ratio is $\mu = \epsilon$, and the approximation ratio achieved by the greedy algorithm is thus $\mu/(1+\mu)$.

However, when μ is a constant, the greedy algorithm achieves a constant performance guarantee. Specifically, we prove the following.

THEOREM 2.4. *The greedy algorithm achieves a performance guarantee of at least* $\frac{(\sqrt{1+\mu}-1)^2}{1+\mu}$.

PROOF. Let $S = \{s_1, \ldots, s_k\}$ denote the set chosen by the greedy algorithm, and $T = \{t_1, \ldots, t_m\}$ an optimal set of minimum size. Note that by the definition of μ, we have

$$g'(T) = \frac{\mu}{1+\mu} g(T).$$

We write $c = 1+\mu$, $y = \sqrt{c} - 1$, $z = 2\sqrt{c} - 1$, $x = y/z$, and $\epsilon = \mu z/c$. Within this notation, we are trying to establish a performance guarantee of y^2/c.

We consider two cases, depending on the relative sizes of S and T. First, suppose $|S| \leq \epsilon |T|$. Since g is a monotone submodular function, we have

$$g(T) \leq g(S \cup T) \leq g(S) + \sum_i [g(S \cup \{t_i\}) - g(S)]. \tag{1}$$

Now when the greedy algorithm stopped, no element had positive marginal gain with respect to g', and hence $g(S \cup \{t_i\}) - g(S) \leq 1$ for each $i = 1, \ldots, m$. Thus, by inequality (1), we have $g(T) \leq g(S) + |T|$, and hence $g'(T) \leq g(S)$. Thus

$$\begin{aligned} \frac{g'(S)}{g'(T)} &= \frac{g(S) - |S|}{\mu |T|} \\ &\geq \frac{\mu |T| - \epsilon |T|}{\mu |T|} \\ &= 1 - \frac{\epsilon}{\mu} = \frac{y^2}{c}. \end{aligned}$$

Otherwise, we have $|S| > \epsilon |T|$. We write $\delta_i = g(\{s_1, \ldots, s_i\}) - g(\{s_1, \ldots, s_{i-1}\})$, with $\delta_1 = g(\{s_1\})$. Note that $g(S) = \sum_{i=1}^{k} \delta_i$. The submodularity of g and the definition of the greedy algorithm implies that $\delta_i \geq \delta_{i+1}$ and $\delta_i \geq 1$ for all i. Also, for the remainder of the analysis, we may assume without loss of generality that $k \leq m$. Indeed, if $k > m$, then the analysis below can be applied instead to the set $\{s_1, \ldots, s_m\} \subseteq S$, showing that $g'(\{s_1, \ldots, s_m\}) \geq (y^2/c) g'(T)$; and by the definition of the greedy algorithm we also have $g'(S) \geq g'(\{s_1, \ldots, s_m\})$.

Let $\ell = \lceil xk \rceil$. If $\delta_\ell \geq 1 + y$, then $g'(S) \geq yxk$, and hence

$$\frac{g'(S)}{g'(T)} \geq \frac{yxk}{\mu m} \geq \frac{yx\epsilon}{\mu} = \frac{y^2}{c}.$$

Otherwise, $\delta_\ell < 1 + y$. Now, if we write $S' = \{s_1, \ldots, s_{\ell-1}\}$, then by analogy with inequality (1) we have

$$g(T) \leq g(S' \cup T) \leq g(S') + \sum_i [g(S') \cup \{t_i\}) - g(S')]. \tag{2}$$

But since S is constructed by the greedy algorithm, we know by our assumption about δ_ℓ that $g(S') \cup \{q\}) -$

$g(S') \leq 1 + y$ for all elements $q \in U$. With inequality (2), this implies

$$g(T) \leq \sum_{i<\ell} \delta_i + m(1+y) \leq \sum_{i<\ell} \delta_i + m(1+y)\delta_j \tag{3}$$

for every $j \in \{\ell, \ell+1, \ldots, k\}$. Summing (3) over all such j, we obtain

$$\begin{aligned}(1-x)kg(T) &\leq (k-\ell+1)g(T) \\ &\leq (1-x)k\sum_{i<\ell}\delta_i + m(1+y)\sum_{i\geq\ell}\delta_i \\ &\leq m(1+y)\sum_{i<\ell}\delta_i + m(1+y)\sum_{i\geq\ell}\delta_i = m(1+y)g(S),\end{aligned}$$

where the final inequality follows since $k \leq m$ and hence $(1-x)k \leq m(1+y)$. Thus we have

$$g(S) \geq \frac{1-x}{1+y} \cdot \frac{k}{m} \cdot (1+\mu)m = \frac{1-x}{1+y} \cdot ck,$$

whence

$$g'(S) \geq k\left[\frac{1-x}{1+y} \cdot c - 1\right]$$

and

$$\frac{g'(S)}{g'(T)} \geq \frac{k}{\mu m}\left[\frac{1-x}{1+y} \cdot c - 1\right] \geq \frac{\epsilon}{\mu}\left[\frac{1-x}{1+y} \cdot c - 1\right] = \frac{y^2}{c}.$$

As a direct corollary of Theorem 2.4, 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 unbounded 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 2.4 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.

3. 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. Recently, Alon and Sudakov [1999] have obtained a polynomial-time approximation scheme for this problem. We nevertheless give our algorithms, in part because two of them run in time linear in n.

The crux of our approach is to show that if we restrict ourselves to policies that are collocated with customers, the loss is modest. Consider any set T of m vertices of the d-cube, $v_1, \ldots, v_m$.

LEMMA 3.1. *Let the hypercube vertex P denote the optimal policy for T. Then there exists a customer v_i such that*

$$\sum_{j=1}^{m} v_i \odot v_j \geq (2\sqrt{2}-2)\sum_{j=1}^{m} P \odot v_j. \tag{4}$$

The bound $(2\sqrt{2}-2) \approx .828$ of Lemma 3.1 cannot be improved significantly; it is easy to construct an example in which no customer gets better than $5/6 \approx .833$ of the benefit of the optimal policy. Consider 3 customers in the 3-dimensional cube, located at the vertices (100), (010) and (001). Clearly the optimal policy (000) results in a benefit of 6, while each of the 3 customer policies yields benefit 5.

Proof of Lemma 3.1: The proof is a counting argument, and will in fact show that

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

Consider an $m \times d$ matrix with v_i's as rows. For $1 \leq r \leq d$, let p_r denote the fraction of 1's in the rth column of this matrix. Assume, without loss of generality, that for all r we have $p_r \geq 1/2$; for otherwise, we can exchange the roles of 0 and 1 as necessary. Consider the left-hand side of (4); the contribution of the rth column is $m^2[p_r^2 + (1-p_r)^2]$. On the other hand, the contribution of the rth column to the right-hand side is no more than $m^2 p_r$. Summing over r and noting that all $p_r \geq 1/2$, we can verify the inequlity (the worst case is $p_r = 1/\sqrt{2}$, for all r). ■

Thus Lemma 3.1 in fact shows that the expected value of a customer policy chosen uniformly at random is at least 0.828 times the value of the optimal policy P.

Let $T_1, \ldots, T_k$ denote the segments of the optimal segmentation; by Lemma 3.1, for each 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 $t_1, \ldots, t_k$ deterministically 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 provide two additional algorithms that run in time linear in the number of customers n. For any set of t customers in d dimensions, the value of the optimal solution is between $td/2$ and td. Let $S_1, \ldots, S_k$ denote the partition of the customers by the optimal segmentation. For any such set S_i, we call it *small* of $|S_i| \leq \epsilon n/(3k)$; else we call it *big*. The net contribution of all the small sets to the objective function is at most $\epsilon nd/3$. Since the value of the optimal solution (which we denote by $\mathcal{V}$) is at least $nd/2$, the contribution of the big sets is at least $(1 - 2\epsilon/3)$ times $\mathcal{V}$.

Our first algorithm picks a random set of k customers as policies, and determines (in time $O(nk)$) the value of the best segmentation for the sample. The algorithm takes the best segmentation from $M = (9/\epsilon)^k \ln(1/\epsilon)$ such trials. Thus, the total running time is $O(nk(9/\epsilon)^k \ln(1/\epsilon))$ steps. We now show that the expected value of this best segmentation is at least $(0.828 - \epsilon)\mathcal{V}$.

Call a sample *good* if it picks at least one customer in every big set. A simple calculation shows that the probability that a sample is good is at least $(\epsilon/3e)^k$. Thus the probability that every one of $(9/\epsilon)^k \ln(1/\epsilon)$ trials is not good is at most $\epsilon^{(9/2e)^k} < \epsilon/3$ for sufficiently small ϵ and/or large k.

Conditioned on a sample being good, each big set gets at least one sample, and this sample is distributed uniformly at random among the customers in that big set. Let the random variable X denote the value of a solution, and X_i the contribution from the ith big set. Let $\mathcal{V}_i$ denote the contribution of the ith big set to $\mathcal{V}$. By Lemma 3.1, $E[X_i | S_i \text{ gets a sample}] \geq 0.828\mathcal{V}_i$. Clearly

$$E[X|\text{ the sample is good}] = E[\sum_i X_i|\text{ the sample is good}] =$$

$$\sum_i E[X_i|S_i \text{ gets a sample}] \geq 0.828 \sum_i \mathcal{V}_i = 0.828(1-2\epsilon/3)\mathcal{V}.$$

Since M trials include at least one good sample with probability exceeding $1-\epsilon/3$, the expected value of the best segmentation from the M trials is at least $(0.828-\epsilon)\mathcal{V}$.

Our second algorithm is simpler in that it only takes a single sample of ℓ customers. We compare the value of this segmentation to $\mathcal{V}$, the best k-segmentation. As before, let $T_1, \ldots, T_k$ denote the segments of the optimal segmentation, and let $\mathcal{V}_i$ denote the value of the ith segment. Let b_i denote $\mathcal{V}_i/\mathcal{V}$, the fractional contribution to $\mathcal{V}$ from the ith segment. For any $i \in [1,k]$, we have $\mathcal{V}_i = \gamma_i |T_i| d$, where $\gamma_i \in [1/2, 1]$. It follows that the probability that a randomly-sampled customer is from T_i is at least $b_i/2$, so that the probability of hitting T_i in ℓ samples is at least $1-(1-b_i/2)^\ell \geq 1-e^{-\ell b_i/2}$. Conditioned on hitting a segment T_i, our expected benefit from that segment is at least $0.828\mathcal{V}_i$. Thus the expected ratio of the value of our random segmentation to $\mathcal{V}$ is

$$\sum_i 0.828 b_i (1-e^{-\ell b_i/2}).$$

This is minimized [3] when all the b_i equal $1/k$, leading to part (3) of the following theorem. We note, however, that our analysis of such a sample is somewhat crude, in that we simply bound every γ_i from below by $1/2$, and thus get a weak bound of $b_i/2$ on the probability of hitting the ith segment. Tightening this analysis remains an interesting direction.

THEOREM 3.2. *The* HYPERCUBE SEGMENTATION PROBLEM *can be approximated as follows:*

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

(2) *Within* $0.828-\epsilon$ *with probability* $1-o(1)$*, by a randomized algorithm running in* $O(nk(6/\epsilon)^k \ln(1/\epsilon))$ *steps.*

(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 *as* ℓ *becomes large.)*

4. META-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 *meta-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 $S \subseteq \mathcal{D}$ is a finite set, 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 [1977] and Nemhauser, Wolsey, and Fisher [1978]; we drew a different but related connection to this work in Section 2.2.

[3] To see this, consider any other vector of values for the b_i; there is at least one greater and one smaller than $1/k$. Move these two values simultaneously towards $1/k$ at the same rate until one of them reaches $1/k$ – as we do so, the ratio decreases. Repeat this construction until all b_i equal $1/k$.

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 with $f_1(x) = v \cdot x$ 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 *meta-submodular* property. We say that a set function g is *meta-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 cast this property in a form that is easier to work with.

LEMMA 4.1. *Let g be a set function. Then g is meta-submodular if and only if for* non-empty *sets $S \subseteq T$ and all elements $x \notin T$, we have*

$$g(S \cup \{x\}) - g(S) \geq g(T \cup \{x\}) - g(T).$$

PROOF. Meta-submodularity implies that $g(S) + g(T \cup \{x\}) \leq g(S \cup \{x\}) + g(T)$, since $S \neq \phi$. To show the converse direction, suppose we are given S and T with $S \cap T \neq \phi$. Write $S \setminus T = \{x_1, \ldots, x_k\}$, $Y_i = (S \cap T) \cup \{x_1, \ldots, x_i\}$, and $Z_i = T \cup \{x_1, \ldots, x_i\}$, with $Y_0 = S \cap T$ and $Z_0 = T$. Then $g(Y_{i+1}) - g(Y_i) \geq g(Z_{i+1}) - g(Z_i)$ by assumption, since each $Y_j \subseteq Z_j$ and (crucially for the case $i = 0$) all Y_i are non-empty. Thus

$$g(S) - g(S \cap T) = \sum_{i=0}^{k-1} g(Y_{i+1}) - g(Y_i) \geq \sum_{i=0}^{k-1} g(Z_{i+1}) - g(Z_i) = g(S \cup T) - g(T),$$

and hence g is meta-submodular.

We now show

LEMMA 4.2. *Every segmentation function σ is meta-submodular.*

PROOF. We use Lemma 4.1. Let S and T be finite subsets of $\mathcal{D}$, with $S \subseteq T$ and S non-empty. Let $x \in \mathcal{D} \setminus T$. For two real numbers p and q, we write $(p - q)^+ = \max((p - q), 0)$. For each $i \in C$, let $a_i^* = \max_{y \in S} f_i(y)$ and $b_i^* = \max_{z \in T} f_i(z)$. Clearly $a_i^* \leq b_i^*$ since the expression for b_i^* involves computing the maximum over a superset of S. Thus

$$\begin{aligned} \sigma(S \cup \{x\}) - \sigma(S) &= \sum_i (f_i(x) - a^*)^+ \\ &\geq \sum_i (f_i(x) - b^*)^+ \\ &= \sigma(T \cup \{x\}) - \sigma(T). \end{aligned}$$

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 is *weakly monotone* if (i) $g(\phi) = 0$; (ii) $g(S) \leq g(T)$ when $S \subseteq T$ and S is non-empty; and (iii) 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* $g(S \cup \{x\}) - g(S)$.

In fact, we will analyze the more general case in which each step of this greedy algorithm is implemented only approximately: an element $x \notin S$ is selected for which $g(S \cup \{x\}) - g(S)$ is within a factor of $1/c$ of the maximum marginal gain, for some parameter c. We will refer to this as the *c-approximate greedy algorithm*.

THEOREM 4.3. *Let g be a set function which satisfies the meta-submodular and weak monotonicity conditions. Then the c-approximate greedy algorithm achieves a performance guarantee of*

$$1 - \left(1 - \frac{1}{ck}\right)^{k-1},$$

which converges to $1 - e^{-1/c}$ from below as k increases.

PROOF. Let $S = \{s_1, \ldots, s_k\}$ denote the set found by the c-approximate greedy algorithm, and let $T = \{t_1, \ldots, t_k\}$ denote the size-k set of maximum value. Let $S_i = \{s_1, \ldots, s_i\}$, $S_0 = \phi$, $\delta_i = g(S_i) - g(S_{i-1})$ (for $i \in \{1, 2, \ldots, k\}$), and $W_i = S_i \cup T$. Now, since $g(\phi) = 0$, we have

$$g(S_i) = \sum_{j=1}^{i} \delta_j,$$

and by the weak monotonicity of f we have

$$g(T) \leq g(W_i).$$

By the meta-submodularity of f and the definition of the c-approximate greedy algorithm, we have

$$g(W_i) \leq g(S_i) + (ck)\delta_{i+1} = \sum_{j=1}^{i} \delta_i + (ck)\delta_{i+1}$$

for $i \in \{1, 2, \ldots, k-1\}$ Thus we obtain the following $k-1$ inequalities:

$$\begin{aligned} g(T) &\leq \delta_1 + (ck)\delta_2 \\ g(T) &\leq \delta_1 + \delta_2 + (ck)\delta_3 \\ \cdots &\leq \cdots \\ g(T) &\leq \delta_1 + \delta_2 + \cdots + \delta_{k-1} + (ck)\delta_k \end{aligned}$$

Suppose we multiply both sides of the i^{th} inequality by

$$\left(1 - \frac{1}{ck}\right)^{k-1-i},$$

and add them all up. Note that the sum of the left-hand-sides is

$$\left(\sum_{i=0}^{k-2} \left(1 - \frac{1}{ck}\right)^{i}\right) g(T)$$

while the sum of the right-hand sides is

$$\left(\sum_{i=0}^{k-2} \left(1 - \frac{1}{ck}\right)^{i}\right) \delta_1 + (ck) \sum_{i=2}^{k} \delta_i.$$

Since $g(\phi) = 0$, and $\delta_1 \geq 0$ by weak monotonicity, we have

$$(ck)g(S) \;=\; (ck) \sum_{i=1}^{k} \delta_i$$

$$\begin{aligned}
&\geq \left(\sum_{i=0}^{k-2}\left(1-\frac{1}{ck}\right)^i\right)\delta_1 + (ck)\sum_{i=2}^{k}\delta_i \\
&\geq \left(\sum_{i=0}^{k-2}\left(1-\frac{1}{ck}\right)^i\right)g(T) \\
&= (ck)\left[1-\left(1-\frac{1}{ck}\right)^{k-1}\right]g(T),
\end{aligned}$$

and hence

$$\frac{g(S)}{g(T)} \geq 1-\left(1-\frac{1}{ck}\right)^{k-1}.$$

As noted above, this converges to $1-e^{-1/c}$ from below as k increases.

Theorem 4.3 thus applies to all weakly monotone segmentation functions. But we observed in Section 2.2 that the implementation of a single step of the greedy algorithm can sometimes be an NP-complete problem in its own right. Thus, this theorem does not directly yield an *efficient* approximation algorithm. A natural approach would be to look for efficient implementations of the c-approximate greedy algorithm for some $c > 1$; we leave this as an open question.

However, Theorem 4.3 does provide an efficient approximation algorithm for segmentation problems arising from optimization over polyhedra that contain the origin and have a polynomial-size set of vertices that can be enumerated in polynomial time. In such a case, the greedy algorithm need only examine vertex solutions, and thus can run in polynomial time; one can also verify that weak monotonicity 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 4.4. *There is an efficient* $[1-(1-1/k)^{k-1}]$*-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-size set of vertices that can be enumerated in polynomial time.*

5. DISCUSSION 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.

Let us present here a final and simple (and, we believe, very compelling) illustration of the relationship between our ideas and the area of clustering. Suppose that a monopoly has n customers, and for each customer i it knows precisely the price elasticity curve of the customer: A linear equation

$$q(p) = a_i - b_i \cdot p$$

giving the quantity q of the commodity customer i would buy if the price were p; a_i and b_i are known constants. The enterprise wants to cluster the customers into k groups, and offer different prices per group so as to maximize revenue. The customers are thus points (a_i, b_i) on the plane, and we are asked to cluster them optimally into k clusters $S_1, \ldots, S_k$, and to choose for each cluster S_j a price p_j (it is easy to see that this price is just

$$p_j = \frac{(\sum_{i\in S_j} a_i)^2}{\sum_{i\in S_j} a_i})$$

so that the total revenue $\sum_{j=1}^{k} \sum_{i \in S_j} a_i p_j - b_i p_j^2$ is as large as possible. We can call this the PRICE SEGMENTATION problem.

Which of the many standard clustering criteria should be adopted for this problem, and which of the many heuristics known for each criterion should we use in order to determine the k segments? As it turns out, in this case the solution is simple and can be found efficiently (we omit the straightforward proof):

THEOREM 5.1. *The optimum clusters in the* PRICE SEGMENTATION *problem are always separated by lines through the origin in the $a_i - b_i$ plane. Thus, the optimum segmentation can be found in $O(kn^2)$ time by dynamic programming (in $O(n^3)$ for variable segmentation.*

Let us emphasize: What is good about this clustering result is not that the clusters are so regular, or that the algorithm is polynomial, or that it gives the precise answer. What is auspicious is that *the right objective is optimized.*

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 1/2-approximation when $k = 2$. Given that we are able to analyze the greedy algorithm at the level of general segmentation problems, it is unfortunate that its *implementation* 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.

Following the preliminary conference version of this work, Alon and Sudakov [1999] obtained several further results on segmentation problems. In particular, they provided a polynomial-time algorithm that, for a fixed number of segments k and a fixed $\epsilon > 0$, produces a solution to the HYPERCUBE SEGMENTATION PROBLEM that is within a $1 - \epsilon$ factor of optimal. They also demonstrate some very strong inapproximability results for segmentated versions of a number of basic problems arising in combinatorial optimization.

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.

Our formulation of the underlying optimization problem faced by an enterprise interested in data mining did not make explicit reference to the notion of competitive environments. However, it is possible to formulate a notion of *segmented matrix games* [Kleinberg et al. 1998], in which each customer has an associated payoff matrix. This leads to a range of further open problems.

REFERENCES

AGRAWAL, R., IMIELINSKI, T., AND SWAMI, A. N. 1993. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, P. Buneman and S. Jajodia, Eds. Washington, D.C., 207–216.

ALON, N. AND SUDAKOV, B. 1999. On two segmentation problems. *J. Algorithms 33,* 1, 173–184.

ARORA, S., KARGER, D., AND KARPINSKI, M. 1995. Polynomial time approximation schemes for dense instances of NP-hard problems. In *ACM Symposium on Theory of Computing*. 284–293.

BERMAN, O., HODGSON, M., AND KRASS, D. 1995. Flow-interception problems. In *Facility Location: A Survey of Applications and Methods*, Z. Drezner, Ed. Springer.

BERN, M. AND EPPSTEIN, D. 1996. Approximation algorithms for geometric problems. In *Approximation Algorithms for NP-Hard Problems*, D. Hochbaum, Ed. PWS.

COGGINS, J. 1983. Dissimilarity measures for clustering strings. In *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*, D. Sankoff and J. Kruskal, Eds. Addison-Wesley.

CORNUEJOLS, G., FISHER, M., AND NEMHAUSER, G. 1977. Location of bank accounts to optimize float. *Management Science 23*, 789–810.

Feder, T. and Greene, D. 1988. Optimal algorithms for approximate clustering. In *ACM Symposium on Theory of Computing*.

Gonzalez, T. 1985. Clustering to minimize the maximum inter-cluster distance. *Theoretical Computer Science 38*, 293–306.

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

Kearns, M., Mansour, Y., and Ng, A. Y. 1997. An information-theoretic analysis of hard and soft assignment methods for clustering. In *Proc. 13th Conference on Uncertainty in Artificial Intelligence*. 282–293.

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

Masan, B. and Piatetsky-Shapiro, G. 1996. A comparison of approaches for maximizing business payoff of prediction models. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*. 195–201.

Nemhauser, G., Wolsey, L., and Fisher, M. 1978. An analysis of the approximations for maximizing submodular set functions. *Mathematical Programming 14*, 265–294.

Piatetsky-Shapiro, G. and Matheus, C. 1994. The interestingness of deviations. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*.

Shmoys, D. B., Tardos, É., and Aardal, K. 1997. Approximation algorithms for facility location problems (extended abstract). In *ACM Symposium on Theory of Computing*. 265–274.

Silberschatz, A. and Tuzhilin, A. 1996. What makes patterns interesting in knowledge discovery systems. *Ieee Trans. On Knowledge And Data Engineering 8*, 970–974.

Smyth, P. and Goodman, R. M. 1991. Rule induction using information theory. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*.

Received November 1999, revised August 2003, accepted August 2003.