TR 00-057

Efficient Algorithms for Creating Product Catalogs

Michael Steinbach, George Karypis, and Vipin Kumar

November 10, 2000

Efficient Algorithms for Creating Product Catalogs*

Michael Steinbach, George Karypis, and Vipin Kumar

University of Minnesota, Army HPC Research Center

{steinbac, karypis, kumar}@cs.umn.edu

Technical Report #00-057

Abstract

For the purposes of this paper we define a catalog to be a promotional catalog, i.e., a collection of products (items) presented to a customer with the hope of encouraging a purchase. The single mailing problem addresses how to build a collection of catalogs and distribute them to customers (one per customer) so as to achieve an optimal outcome, e.g., the most profit. Each catalog is a subset of a given set of items and different catalogs may contain some of the same items, i.e., catalogs may overlap. A slightly more general, but important extension of the single mailing problem seeks the optimal set of catalogs when multiple mailings are allowed, i.e., multiple catalogs can be sent to each customer. Catalog creation has important applications for e-commerce and traditional brick-and-mortar retailers, especially when used with personalized recommender systems.

The catalog creation problem is NP complete and some (relatively expensive) approximation algorithms have recently been developed. In this paper we describe more efficient techniques for building catalogs and show that these algorithms outperform one of the previously suggested approaches. (Indeed, the techniques previously suggested are not feasible for realistic numbers of customers and catalogs.) Some of our techniques directly use the objective function, e.g., maximize profit, to find a locally optimal solution in an approach based on gradient ascent. However, by combining such techniques with the clustering of similar customers, better results can sometimes be obtained. We also analyze the performance of our algorithms with respect to a theoretical bound and show that, in some cases, their performance is close to optimal.

1 Introduction

Recent years have seen a sharp increase in the amount of customer information being gathered and warehoused by commercial enterprises. The large quantities of historical customer data and the emergence of e-commerce have motivated a great deal of research into personalization technology. In particular, collaborative filtering based recommendation engines have become popular for providing highly "personalized" information by identifying, for each customer, a small set of items that will be of interest to that customer. For example, an online bookseller might use a recommender system to create, in real time, a list of "related" or "recommended" books for the basket of books that a customer purchases. However, recent work [KPR98], has questioned the need for such complete personalization on the grounds of efficiency and accuracy. Also, even if accuracy issues are ignored, traditional brick and mortar retailers can only personalize to a limited extent, as only a relatively small number of different product catalogs can be created. Thus, in this paper we use information about expected customer buying behavior (from recommender engines, statistical techniques, or elsewhere) as our starting point, but adopt the view that often the most profitable use of this information is the creation of a relatively small number of "group targeted" catalogs, i.e., catalogs not personalized to the individual level but to a like-behaving group of customers. To that end, we seek to develop efficient catalog creation algorithms, where we define a catalog to be a promotional catalog, i.e., a collection of products (items) presented to a customer with the hope of encouraging a purchase.

The conceptual foundation for our work is the microeconomic framework for data mining that was introduced in [KPR98]. In this framework, a retailer attempts to choose the business decision that maximizes the expected profit. We say that profit is the objective function to be maximized. If all the customers contribute independently to the profit, then the total profit is just the sum of the profits resulting from each customer as a result of the business decision. If our business decision involves deciding which set of items (catalog) to present/promote to a customer, then resulting profit will be the sum of the profits of each item that a customer buys from that catalog. Making a single business decision for all customers does not yield the maximum profit and it is advantageous to consider the *segmented* version of this optimization problem. In other words, we choose a set of *k* business decisions such that if each customer is assigned the most profitable business decision, then the overall profit is maximized. (Notice that finding the best *k* decisions implicitly divides or segments the customers into *k* groups.) In the optimal case, we would have a separate decision for each customer, but typically this is impractical, and the customers are segmented

* This work was supported by NSF CCR-9972519, EIA-9986042, ACI-9982274, by Army Research Office contract DA/DAAG55-98-1-0441, by the DOE ASCI program, and by Army High Performance Computing Research Center contract number DAAH04-95-C-0008.

into a small number of groups. In terms of catalogs, this segmentation corresponds to dividing the customers into groups, each of which gets a different catalog.

In [KPR98] it is shown that segmentation problems are NP complete, in general, and various approximation algorithms were introduced to solve these problems. The various algorithms proposed in [KPR98] are based on sampling followed by an exhaustive enumeration of all possible segmentations. Unfortunately, as our experiments show, in order to achieve high quality results, these approaches require large sample sizes, making them expensive for any realistic catalog creation problem.

In this paper we present computationally efficient algorithms for solving two different catalog creation problems, the ***single mailing*** and the ***multiple mailings*** problems. The single mailing problem addresses how to segment the customers and send each segment a different catalog, while the multiple mailings problem addresses the situation where each customer gets multiple catalogs over time. The single and multiple mailings problems have a number of applications for traditional brick-and-mortar retailers and mail-order companies, as well as for new *e*-commerce companies. In particular, the solution to the single-mailing problem will allow retailers to automatically build the right set of catalogs targeted to different segments of their customer base. The multiple-mailings problem allows retailers to plan their marketing campaign over a period of time. In the case of *e*-tailers, both the single and the multiple-mailing problem can be used to build a small number of targeted "mini-catalogs" that can either be sent directly to the customers or used to create "banner-ads" to be displayed by the various banner-ad companies.

We developed three different algorithms for solving the single-mailing problem. The first algorithm, called ***indirect catalog creation***, uses a clustering algorithm to identify the customer segments and then derives the best catalog for each segment. The second algorithm, called ***direct catalog creation***, tries to simultaneously identify both a catalog and its associated customer segment. Finally, the third algorithm, called ***hybrid catalog creation***, solves the problem by combining elements of the earlier two algorithms. Our algorithms for the multiple mailings problem are based on iteratively applying the various algorithms for the single-mailing problem by appropriately modifying the input data after each iteration. We experimentally evaluated the performance of our algorithms on two different datasets corresponding to the historical purchasing transactions of a mail-order company and an *e*-retailer. Our experiments show that the proposed algorithms significantly outperform the algorithms based on exhaustive enumeration, and produce results that are close to the optimal.

The rest of this paper is organized as follows. Section 2 provides a formal definition of the two catalog creation problems that are the focus of this paper and offers some application examples in which they apply. Section 3 describes some of the existing algorithms that can be used to solve certain restricted instances of these problems. Section 4 describes the different algorithms that we developed for solving the single mailing and multiple mailings problems. Section 5 provides a detailed experimental evaluation of our algorithms on two commercial datasets. Finally, Section 6 offers some concluding remarks.

2 Problem Definition

The purpose of this paper is to develop efficient algorithms for automatically building product-catalogs in two different settings. The input to the proposed algorithms is an $n \times m$ customer-item matrix P, where n is the number of customers, m is the total number of distinct products, and $p_{i,j}$ represents the profit that will be achieved from the jth item belonging to the ith customer.

Given the above definitions, the first catalog creation problem is stated as follows:

Select k sets of items $\{I_1, I_2, \ldots, I_k\}$, each containing no more that q items, and assign exactly one of these subsets to each one of the customers so that $\sum_{i=1}^{n} \sum_{j \in S(i)} p_{i,j}$ is maximized, where *S(i)* is the set of items assigned to the ith customer.

Essentially, the problem is that of segmenting the customers into k groups, and selecting a set of items (*i.e.,* a product catalog) to send to each one of these segments so that the overall potential profit is maximized. Note that this problem formulation assumes that each customer can only buy products that are contained in the catalog that is assigned to him/her, as he/she will not be aware of the other products offered by the retailer. We will refer to this problem formulation as the ***single mailing*** problem.

There are two key parameters in the above problem formulation. The first is the number of catalogs (k), and the second is the maximum number of products (q) that is included in each catalog. This parameterization was motivated for the following reasons. First, in many applications areas (*e.g.,* traditional mail-order companies), the cost of developing and printing different catalogs is considerably higher than the cost of printing the same catalog more times. Thus, keeping k small can reduce marketing costs. Second, due to physical constraints, mailing costs, and limited customer attention span, limiting the number of products in each catalog quite often increases the effectiveness of a marketing campaign. For example, in the e-commerce domain even e-mailed "mini-catalogs" tend to be relatively small, sometimes for physical reasons, but often because customers only look at a small number of items. As another example, the number of items in banner ads are restricted by the amount of space available on a Web page.

The second catalog creation problem generalizes the above formulation to the case in which we can send multiple catalogs to each customer. The problem is stated as follows:

> Select l times k sets of items $\{I_{1,1}, I_{1,2}, \ldots, I_{1,k}\}, \ldots, \{I_{l,1}, I_{l,2}, \ldots, I_{l,k}\}$, each containing no more that q items, and assign one from each k-subset to each of the customers so that $\sum_{i=1}^{n} \sum_{j \in S(i)} p_{i,j}$ is maximized, where *S(i)* is the set of items assigned to the i^{th} customer.

Essentially, this formulation computes l distinct instances of the single-mailing problem, but the overall profit is determined only with respect to the set of $l * q$ products that are contained in the l catalogs received by each customer (and not the total $k * l * q$ products that are contained in all the different catalogs). We will refer to this problem as the ***multiple mailings*** problem.

Conceptually, multiple catalogs per customer address the problems of limited customer attention and the limited number of items in a catalog. Up to some limit, a new catalog does get a customers attention, and a new catalog can contain items that were omitted in previous catalogs. Multiple catalogs per customer also address the issue that customers are (hopefully) unwilling to spend more than a reasonable amount of money at any one time.

The above catalog creation problems are combinatorial optimization problems involving the total number of catalogs, k, the number of catalogs sent to each customer, l, the number of items in each catalog, q, and which items are in which catalog. A formal foundation for the basic catalog problem is provided by the microeconomic framework for data mining that was introduced in [KPR98]. However, the multiple mailings problem was not discussed in that paper.

3 Related Work

In this section we quickly survey a couple of algorithms for solving special cases of the single mailing problem. The algorithm for solving the single mailing problem will be used in all of the following algorithms.

3.1 The Single Mailing Problem with One Catalog

If we want only a single catalog for a group of customers, then we can find the optimal catalog in a very straightforward way. In particular, the optimal catalog for a group of customers, C, can be found by sorting the items according to expected profit and taking the q most profitable items as the catalog. Note that this procedure is just the fractional Knapsack algorithm [CLR90].

Algorithm for the Single Mailing Problem with One Catalog

```
1. For each item, $j$, calculate the total expected profit by summing over all customers, i.e., $\sum_{i \in C} p_{i,j}$ .
2. Sort the items by total expected profit.
3. Select the most profitable $q$ items as the catalog.
```

3.2 The Single Mailing Problem with Two Catalogs

In [KPR98] an algorithm was outlined for solving the catalog creation problem for two catalogs and a bound was given on the algorithm's performance. The initial step of this algorithms is to take a random sample of customers, where the sample size is size proportional to log(n). The algorithm then enumerates all possible splits of this customer sample into two disjoint groups, and, for each split, determines the best pair of catalogs by using the one catalog algorithm. Finally, the optimal set of catalogs is determined by selecting the pair of catalogs with the highest profit for the set of all customers. (Notice that the optimal catalogs can also be viewed as an optimal partition of the customers.)

Exhaustive Algorithm for the Single Mailing Problem with Two Catalogs
1. Take a sample of size, T=c*log(n), customers, where c is some constant. 2. Determine all possible partitions of the customers into two groups. (Note that there are 2^T such partitions.) 3. Determine the best pair of catalogs for each partition. 4. For each set of catalogs derived from a particular partition, assign each customer (from the set of all customers) to the best catalog and calculate the total profit. 5. The optimal set of catalogs is the set with the highest profit.

It was proven that this algorithm would work well provided that the sample size is large enough and all partitions of the sample are enumerated. This algorithm can also be extended to handle more than two catalogs although the details of that extension were not given.

4 Algorithms for the Single and Multiple Mailings Problems

4.1 Single Mailing Problem

The close relationship between the problem of building product catalogs and that of clustering the customers into related groups lead us to explore clustering-based techniques for solving the single mailing and multiple mailings problems. These algorithms are described in the rest of this section.

4.1.1 Indirect Catalog Creation Algorithm

As discussed in Section 3.1, the problem of building a single product catalog for a group of customers can be solved in an optimal way by selecting the q most profitable products. Motivated by this observation, we developed an algorithm that builds the catalogs by using a two-step approach. In the first step we partition the customers into k disjoint sets, and in the second step, we use the "one catalog algorithm" to find the optimal q items for each set. Since the product catalog is determined as a by-product of the customer partitioning, we will refer to this as the ***indirect catalog creation*** **(ICC)** algorithm.

The overall quality of the resulting set of catalogs is highly dependent on the way the different sets of customers are formed. In general, we would like each set to contain similar customers. For this reason, our algorithm uses a clustering algorithm [DJ88, KR90] to group together similar customers. For efficiency reasons, we use a clustering algorithm based on K-means. In the rest of this section we describe the method that we use to represent the customers and the details of the clustering algorithm itself.

Our representation of customers is based on the vector-space model used widely in the field of information retrieval [Rij79, Kow97]. In this model, a vector in the item-space represents each customer, i.e., the j^{th} component of the vector represents the profit associated with the j^{th} item. Thus, the i^{th} customer is represented by a vector, $\boldsymbol{c}_i = (p_{i,1}, p_{i,2},\ldots,p_{i,m})$. The similarity between two customers vectors $\boldsymbol{c}_1$, and $\boldsymbol{c}_2$ is measured using the *cosine* function defined as:

$$cosine(\boldsymbol{c}_1, \boldsymbol{c}_2) = (\boldsymbol{c}_1 \bullet \boldsymbol{c}_2) / \|\boldsymbol{c}_1\| \, \|\boldsymbol{c}_2\|;$$

where "$\bullet$" indicates the vector dot product and $\|\boldsymbol{c}_i\|$ is the length of vector $\boldsymbol{c}_i$.

The key feature of the ICC algorithm is the algorithm used to cluster customers. Over the years a variety of clustering algorithms have been developed with varying time-quality trade-offs [CKPT92, LA99]. Recently, partitional based clustering algorithms (*e.g., K*-means and its variants) have gained widespread acceptance for sparse, high dimensional data sets as they provide reasonably good clusters and have a near-linear time complexity

[CKPT92, LA99]. For this reason, the clustering algorithm used by ICC is derived from this general class of partitional algorithms.

Partitional clustering algorithms compute a *K*-way clustering of a set of customers either directly or via recursive bisection. A direct *K*-way clustering is computed as follows. Initially, a set of *k* customers is selected from the collection to act as the *seeds* of the *k* clusters. Then, for each customer, its similarity to these *k* seeds is computed, and it is assigned to the cluster corresponding to its most similar seed. This forms the initial *K*-way clustering. This clustering is then repeatedly refined using the following procedure. First, the centroid vector for each cluster is computed, and then each customer is assigned to the cluster corresponding to its most similar centroid. This refinement process terminates either after a predetermined small number of iterations, or after an iteration in which no customer moved between clusters. A *K*-way partitioning via recursive bisection is obtained by recursively applying the above algorithm to compute a 2-way clustering, *i.e.*, bisections. Initially, the customers are partitioned into two clusters, then one of these clusters is selected and is further bisected, and so on. This process continues *k*-1 times, leading to *k* clusters. A number of different schemes have been developed for selecting the initial set of *seed* customers [CKPT92, LA99]. A commonly used scheme is to select these seeds at random. In such schemes, a small number of different sets of random seeds are often selected, a clustering solution is computed using each one of these sets, and the best of these solutions is selected as the final clustering.

Our experiments as well as related experiments in the context of clustering document datasets [SKK00] have shown that bisecting *K*-means produces better results than those obtained by the direct *K*-means algorithm. For this reason, in ICC we compute a *K*-way clustering using the bisecting approach. For bisecting *K*-means, a number of different ways have been proposed for selecting the cluster to split next in the sequence of the *k*-1 bisections [KE00, SKK00]. For the ICC approach we found that the approach that splits the largest cluster produces the best results and this is the scheme that we used. Finally, our clustering algorithm uses the random seed approach, and selects the best (lowest squared error) solution obtained out of five random sets of seeds.

4.1.2 Direct Catalog Creation Algorithm

One of the potential limitations of the ICC algorithm is that there is a mismatch between the objective function used to find the customer clusters and the overall objective of maximizing the profit derived from the catalogs. To illustrate this, consider the very simple case in which we want to build two catalogs, each having a single product from the profit matrix shown in Figure 1. The *K*-means clustering algorithm will put the top four customers into one cluster and the rest into the second cluster, and will build the catalogs {I2} and {I6} for each cluster, respectively. However, the catalogs {I1} and {I5} lead to the highest profit. The reason for the performance difference is that the ICC algorithm does not take into account the size, *i.e., q* of the desired catalogs, but tries to group customers together assuming that *q = m,* the total number of items. In fact, the clusters in the previous example, yield the best *q* = 3 catalogs.

	I1	**I2**	**I3**	**I4**	**I5**	**I6**	**I7**	**I8**
C1	5	4	3	3				
C2	5	4	3	3				
C3		4	3	3	5			
C4		4	3	3	5			
C5	5					4	3	3
C6	5					4	3	3
C7					5	4	3	3
C8					5	4	3	3

Figure 1: An Example Profit Matrix

For this reason we developed a new algorithm that is similar in spirit to the *K*-means algorithm described in the previous section, but directly optimizes the exact objective function. The key difference between the ICC algorithm and the new algorithm is that it directly finds the catalogs and their associated customer segments all in one step. For this reason, we refer to this as the ***direct catalog creation*** **(DCC)** algorithm. The general outline of the DCC algorithm is as follows:

```
Direct Catalog Creation Algorithm
    1. Find an initial set of $k$ catalogs each containing $q$ items.
    2. Form $k$ groups of customers by assigning each customer to the catalog that gives the best
       profit.
    3. Calculate the optimal catalog of $q$ items for each group.
    4. Repeat steps 2 and 3 until there is no change or for a predetermined number of iteration.
```

During the initialization phase (step #1), a set of *k* catalogs are created. These initial catalogs can be created in a number of ways. One-way is to randomly partition the customers into *k* groups, and use the algorithms described in Section 2.1 to build the optimal catalog for each group. An alternate way is to randomly select *k* customers and use the *q* most profitable items from each customer. The former approach is what we are using in our experiments. Note that the latter approach is very similar in spirit to the random *seed* selection scheme used by the *K*-means algorithm.

During the second step of the algorithm each of the customers is re-assigned to the catalog that generates the highest profit. Once this new segmentation of customers has been computed, the optimal catalog for each cluster is computed, and this becomes the current solution for the single mailing problem. The entire process continues for a fixed number of iterations or until the algorithm converges.

The DCC algorithm can build all *k* catalogs in a single step or via a sequence of *k*-1 bisections. As was the case with the *K*-means clustering algorithm, the bisecting variant of the DCC algorithm achieved somewhat better performance, and was used in the experiments reported in Section 5. However, the choice of the cluster to bisect is driven by profit, i.e., conceptually we do a trial bisection of all current clusters and then actually bisect the cluster that leads to the highest profit. (In practice this can be done efficiently by saving the results of previous bisections and, at each step, only bisecting the two new clusters resulting from the previous step.) Finally, the DCC algorithm is executed multiple times (5 in our experiments) using different initial sets of catalogs, and the run that leads to the highest profit is selected as the final solution.

4.1.3 Hybrid Algorithm

Even though the DCC algorithm, by directly maximizing the overall profit, can achieve better solutions than the ICC algorithm, it may be more susceptible to being trapped in local minima than the *K*-means algorithm used by ICC. The reason for this is that during the various iterations of the algorithm, the assignment of each customer to a particular catalog is based only on the profit derived from this customer with respect to the *q* items that make up this catalog, and it ignores how well this customer matches the rest of the cluster's customers in terms of non-catalog products. A consequence of this is that quite often the customers assigned to each catalog have very few items in common other than the catalog defining items. As a result, the iterative refinement performed by DCC will quickly get trapped into local minima.

To address this problem we developed an algorithm that combines elements of both the indirect and direct catalog creation algorithms. Like the DCC algorithm the choice of which cluster to bisect is driven by the profit. However, like ICC, the actual bisection of a cluster is performed using K-means. Also, the HCC algorithm is, as with ICC and DCC, executed multiple times (5 in our experiments) using different initial sets of catalogs, and the run that leads to the highest profit is selected as the final solution. Finally, note that the catalog solution produced in this way becomes the set of initial catalogs for the DCC algorithm. Essentially, the DCC algorithm acts as a post-processing or "refinement" step that improves the catalogs. We will refer to this as the ***hybrid catalog creation*** **(HCC)** algorithm since it uses both the true objective function (profit) and the indirect objective (put similar customers together).

This hybrid algorithm holds the promise of addressing the shortcomings of the individual algorithms. By using the DCC algorithm to refine the catalogs computed by the initial steps of the HCC algorithm, we always generate a better solution, if possible, since the final solution optimizes the real objective function. Furthermore, since the *K*-means solution generated by the *K*-means bisection step of HCC, is a good solution for the catalog creation problem in the hypothetical $q = m$ case, the DCC algorithms starts at a reasonably good solution, and thus, it is not as susceptible to bad local minima.

4.1.4 Summary of Single Mailing Algorithms

The differences between the three algorithms (ICC, DCC, and HCC) that we developed for solving the single-mailing problem are summarized in Table 1. There are three different parameters along which the algorithms differ. The first parameter is the objective function used during the bisection of a customer segment. In the case of the ICC and HCC algorithms, the bisection was done so that it optimizes the K-means objective function whereas in the case of the DCC algorithm, the bisection was performed so that it optimizes the aggregate profit achieved by the catalogs for the two resulting customer segments. The second parameter is the method used to select which segment to bisect next. In the case of ICC, the largest segment was selected, whereas in the case of DCC and HCC, the segment whose bisection leads to the highest profit is selected. Finally, the third parameter has to do with whether or not the final solution is further refined using a *k*-way refinement algorithm. In the case of ICC no such refinement is performed and in the case of DCC and HCC are refinement iteration is performed that optimizes the overall profit of the different catalogs.

	Method Used to Bisect a Cluster	**Criterion to Select which Cluster to Bisect**	**Post-Processing Technique**
ICC	K-means	Size	None
DCC	Profit	Profit	Profit
HCC	K-means	Profit	Profit

Table 1: Comparision of ICC, DCC and HCC.

Finally, the computational complexity of ICC is O(nlog(*k*)), where *n* is the number of customers. The complexity of DCC and HCC is somewhat higher due to the post-refinement iteration and it is O(n*k*).

4.2 Multiple Mailings Problem

The algorithms for solving the single mailing problem increase profit by sending a different catalog to different segments of the customer population, thus increasing the number of items "covered." The multiple mailings approach allows each customer to receive multiple catalogs. Intuitively this should also increase the number of items "covered," and thus result in greater profit. Note that this might not mean that there are more distinct items overall, just that each customer sees more relevant items.

Recall that the multiple mailings problem is to generate $k * l$ catalogs such that each customer receives exactly l catalogs and each catalog contains q items. A naïve approach to solving this problem would be to generate k catalogs, each with $l * q$ items, and then split each of the resulting k catalogs into l separate catalogs of q items to obtain our final set of $k * l$ catalogs. In other words, we build a smaller number of large catalogs and split them to get the desired number of small catalogs per person. However, this approach creates the additional constraint that the same segments of customers will receive the same set of l catalogs, leading to sub-optimal solutions and loss in potential profit.

A better approach for solving the multiple mailings problem is to take an incremental approach. We first apply an algorithm to find the best k catalogs of q items. (Any algorithm that solves the single mailing problem can be used with this algorithm; we will use the best of the single mailing algorithms.) Then, for each customer, we eliminate any items that are in the customer's assigned catalog. We repeat this process for l steps, yielding $k * l$ catalogs. With this algorithm each customer is not restricted to stay in the same segment with the same customers, but will potentially be in l different segments, one for each iteration of the algorithm. Furthermore, by eliminating products that a customer could have purchased from an earlier catalog, we find catalogs that include different products, thus, maximizing profit.

Multiple Mailings Algorithm

```
1. Repeat steps 2 and 3 $l$ times.
2. Find $k$ catalogs of $q$ items.
3. For each customer, $i$, associated with a given catalog, $j$, set $p_{i,j} = 0$.
```

5 Experimental Results

5.1 Data Sets

In this section we compare the performance of the algorithms described in the previous section. We will use two data sets that represent real data from a large commercial retailer. The first data set, TRAD, was derived from three years of traditional mail order catalog sales from Fingerhut Corporation, a large catalog retailer. There are 7,815 customers and 23,554 different items. The second data set, ECOM, was derived from one year of Web-based sales. There are 6,668 customers and 17,491 different items.

5.2 Experimental Results for Catalog Creation Algorithms

5.2.1 Algorithms for the Single Mailing Problem

In this section we compare the three algorithms for the single mailing problem that we discussed in Section 4.1: the indirect catalog creation algorithm (ICC), the direct catalog creation algorithm (DCC), and the hybrid catalog creation algorithm (HCC).

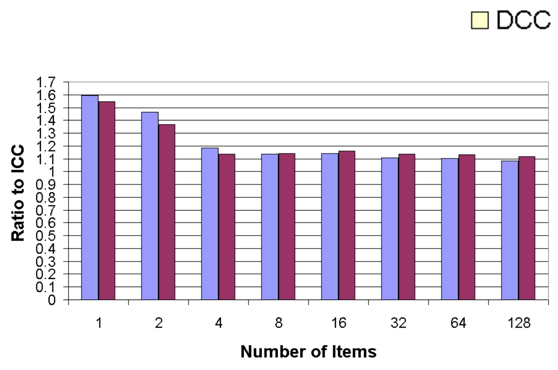


Figure 2a: Ratio of ICC Profit to DCC and HCC profit for 16 catalogs for the ECOM dataset.

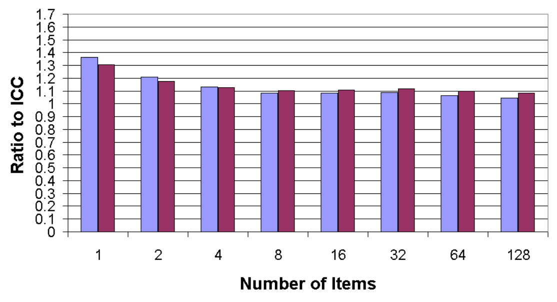


Figure2b: Ratio of ICC Profit to DCC and HCC profit for 64 catalogs for the ECOM data set.

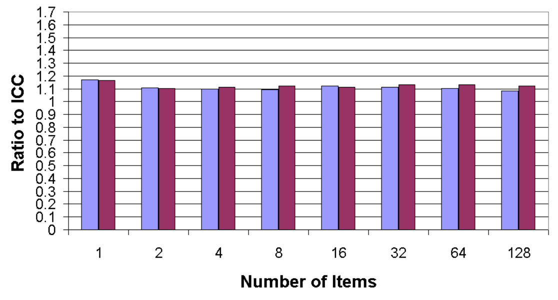


Figure 2c: Ratio of ICC Profit to DCC and HCC profit for 16 catalogs for the TRAD data set.

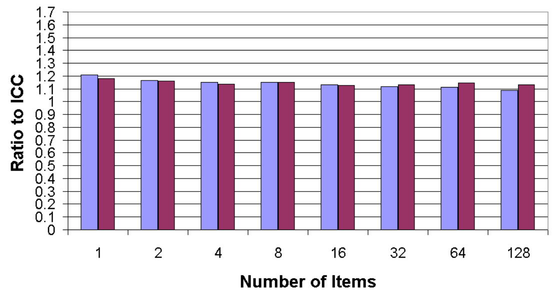


Figure2d: Ratio of ICC Profit to DCC and HCC profit for 64 catalogs for the TRAD data set.

Figures 2a and 2b show the DCC and HCC catalog profit for the ECOM dataset for 16 and 64 catalogs, respectively, as a ratio of the ICC profit, whereas Figures 2c and 2d show the similar set of results for the TRAD dataset. Looking at the results in these figures, we can make a number of interesting observations. First, for all sets of experiments the profit achieved by either the DCC or the HCC algorithms is higher than the profit archived by the ICC algorithm (all the bars are above the 1.0 line). The average improvement achieved by DCC over all 32 experiments was 15.3%, while the average improvement for HCC was 15.6%. These results should not be surprising, as both the DCC and HCC algorithms optimize the true objective function. Second, the improvements achieved by DCC and HCC appear to be higher for catalogs that have fewer items. As discussed in Section 4.1.2, we believe this is due to the fact that the ICC algorithm looks at the entire set of items when trying to cluster the customers and not just the few items that will make up the catalog. As a result, when the number of items is small, the resulting customer segmentation does not lead to high-quality catalogs, but as the number of items gets larger, the clustering solution tends to improve. Third, comparing DCC against HCC, we can see that in general, for catalogs with a small number of items, the DCC algorithm outperforms the HCC algorithm, but as the number of items increases, the HCC becomes somewhat better than DCC. We believe, as discussed in Section 4.1.3, this is due to the fact that the DCC tends to get trapped in local minima since it only considers the items in the current catalogs when refining customer segments. On the other hand, the *K*-means algorithm used by HCC to compute the initial segmentations does not suffer from this problem. However, for small numbers of items, the substantial benefits obtained by directly focusing on the catalog items far outweigh the potential losses due to local minima. For this reason, DCC does better than HCC for small catalogs. However, the relative performance differences between DCC and HCC are quite small.

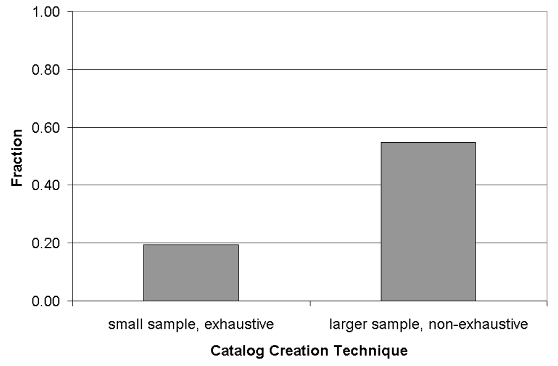
5.2.2 Sampling and Exhaustive Enumeration

We briefly mention the results of some tests with the "exhaustive" algorithm proposed by [KPR98] that was described in section 3.2. For the ECOM data set we sought 2 catalogs of 16 items each. We tried three techniques: exhaustive enumeration of the best catalogs using a sample of 14, partial enumeration of the best catalogs of a larger sample of size 100, and HCC. The partial enumeration technique takes a sample of size 100, but only selects (randomly) 4096 of the 2^{100} possible splits.

The results are shown in Figure 3, which is a relative comparison of the first two algorithms to the hybrid approach. From these results we can see that in both experiments the HCC performs substantially better. In particular, HCC achieves over five times higher profit than the exhaustive enumeration approach, and about 1.8 times higher profit than the partial enumeration approach.

We also did other experiments whose results are not reported here. The only time that we saw reasonably good results was when the number of items was quite low. Thus, while sampling and enumeration can theoretically produce good results, for any realistically sized data sets, either all partitions of the sample can be exhaustively enumerated, but the sample is too small to be representative, or the sample is large enough to be representative, but all possible partitions of the sample cannot be exhaustively enumerated. Thus, techniques based on sampling and enumeration is expensive and/or unlikely to produce good solutions.

Figure 3: Results from a small sample, exhaustive enumeration technique and a larger sample, non-exhaustive enumeration technique, as a fraction of HCC for the ECOM data set (2 catalogs, 16 items).



5.2.3 Multiple Mailings Results

Table 2 shows several solutions for the multiple mailings problem which were produced by running the multiple mailings algorithm of Section 4.2. For all cases there are 32 items per customer. (This is the total item count per customer over all catalogs that the customer receives.)

The first line corresponds to a single mailing solution with a total of 32 catalogs each having 32 items. From this solution, we can obtain a variety of solutions to the multiple-mailings problem by splitting each catalog of 32 items into a larger number of smaller catalogs (*i.e.,* using the naïve approach described in Section 4.2). For example, we can obtain a solution in which *k*=32, *l*=2, and *q*=16, by simply splitting each one of the original catalogs into two smaller catalogs each having only 16 items. However, the total profit for each of those solutions would be the same.

The remaining lines of Table 2 show the solutions of different multiple-mailings problem instances obtained using our multiple mailings algorithm described in Section 4.2. For instance, the second line corresponds to the solution of the problem in which *k*=32, *l*=2, and *q*=16, the third line corresponds to the solution of the problem in which *k*=32, *l*=4, and *q*=8, and so on. Note that the different instances of the multiple mailings problem in Table 2 were selected in such a way (*k* was kept fixed, *l* was increased while *q* was decreased) so that we can compare the results against those obtained by the naïve algorithm (first row of the table). Note that the last column of Table 2, shows the profit achieved by our multiple mailings algorithm relative to that achieved by the naïve algorithm.

k Catalogs (per iteration)	*l* Catalogs per customer (iterations)	*q* Items per Catalog	Profit	Ratio to Naïve
32	1	32	1586931	1.00
32	2	16	1756678	1.11
32	4	8	1892627	1.19
32	8	4	1980063	1.25
32	16	2	2055711	1.30
32	32	1	2101629	1.32

Table 2: Solutions produced by the multiple mailings algorithm compared to the naive approach.

From the results in Table 2, we can see that our algorithm finds catalogs that achieve considerably better results (as measured by the overall profit) compared to those achieve by the naïve algorithm. Moreover, as *q* increases, the performance difference also increases. This was expected because, as discussed in Section 4.2, our algorithm by computing different customer segmentations for each of the different mailings is able to better optimize the overall profit.

An interesting observation that can be made from the results in Table 2, is that as we increase the number of catalogs *sent* to each customer (by increasing *q*) we achieve higher profit even thought the total number of items *seen* by each customer is the same. The reason for that is that by increasing *q* we are able to achieve a higher degree of personalization; thus, we are able to better target the different customers. However, the rate of improvement in profit tends to decrease as *q* increases, indicating that multiple catalogs are quite effective in personalizing the *marketing campaign* and after some point, the additional improvements achieved by better personalization are only marginal. This effect is important because there is always a trade-off between the degree of personalization and the cost associated with highly personalized marketing campaigns. The different solutions to the multiple mailings problem can be used to find the right balance between and optimize the overall *utility* of the marketing campaign.

5.3 Evaluation of Catalog Creation Algorithms With Respect to Optimal Performance

In this section we examine how close our algorithms come to optimal performance. This is not a question that we can fully answer, although the examination of theoretical bounds is helpful. Eventually we also hope to use models of customer data to generate artificial data where the optimal catalogs are known and then evaluate our algorithms on this data.

Consider a set of *k* catalogs, each with *q* items. Can we find an upper bound on the profit achieved by our catalogs? If so, then we could express the performance of our algorithms as a percentage of this bound. There is such a bound, albeit not a very tight one, at least for some cases.

We can find a bound on the profit of *k* catalogs, each with *q* items, by constructing a catalog of *k* * *q* items and using the profit of this catalog as our bound. To see this, consider putting *k* catalogs together to form a single catalog of *k* * *q* items. The profit of this combined catalog must be at least as much as the previous *k* catalogs since each customer sees all the "old" items, as well as many new ones. Thus, this combined catalog can be no better than the optimal catalog of $k * q$ items.

Table 3 shows how this bound can be applied to a collection of catalogs. (We are using the ECOM data set here.) Each entry in the table represents a set of catalogs where the total number of all items in all catalogs is 32. However, this is the only constant, and all of the other quantities vary: the number of catalogs, the number of catalogs per customer, and the number of items per catalog. The important idea to keep in mind is that in all cases, putting together all the catalogs yields a combined catalog of 32 items, and thus, we can bound the results for any of these collections of catalogs by the profit of the optimal catalog for 32 items.

k Catalogs (per iteration)	*l* Catalogs per customer (iterations)	*q* Items per Catalog	Total profit	Ratio to Bound
1	1	32	411,466	1.00
1	2	16	411,466	1.00
1	4	8	411,466	1.00
1	8	4	411,466	1.00
1	16	2	411,466	1.00
1	32	1	411,466	1.00
2	1	16	391,199	0.95
2	2	8	404,520	0.98
2	4	4	404,928	0.98
2	8	2	407,978	0.99
2	16	1	408,877	0.99
4	1	8	380,040	0.92
4	2	4	394,699	0.96
4	4	2	403,725	0.98
4	8	1	405,746	0.99
8	1	4	367,925	0.89
8	2	2	390,309	0.95
8	4	1	400,971	0.97
16	1	2	358,594	0.87
16	2	1	387,873	0.94
32	1	1	361,539	0.88

Table 3. Profit as Fraction of Bound for Different Sets of Catalogs with a Total of 32 items.

The first six entries are really just different instances of 1 catalog of 32 items, (possibly split into multiple catalogs) which is sent to all customers. This catalog represents the optimal profit for a maximum of 32 items. The profit for all other cases shown must be less than this bound, although we emphasize that the true optimal bounds for these other cases may be less than that of the theoretically optimal bound for 1 catalog of 32 items. In other words, the fractions in Table 3 represent worst-case bounds on how close our catalog solutions are to the optimal solution of each case.

Looking at the results in Table 3, we can see that the multiple mailings algorithm performs quite well. In general, the different solutions are within 10% of the upper bound on the potential profit. Furthermore, since some of the cases in Table 3 represent sets of catalogs with just 1 catalog per customer, we can also conclude that our single mailing algorithm is working well for these cases.

Unfortunately, for catalog configurations with larger numbers of catalogs and items, this bound is no longer so tight. For example, for the ECOM data set, if we have $k = 16$, $l = 2$, $q = 16$, then the profit of the catalogs produced by our multiple mailings algorithm is 1,586,930 or 70% of the bound of 2,249,918, which is the maximum for one catalog of 512 items. It is likely that the optimum profit for this case is less than the bound of 2,249,918, but we can no longer accurately judge how well our catalog creation algorithm is performing.

We caution that it can be difficult to make a tradeoff between the total number of catalogs, the number of catalogs per customer, the number of items per catalog and the number of items per customer. Solutions from the multiple mailings algorithm can be combined with other business information to help answer that question.

6 Conclusions

The presentation of promotional product information to customers via the Web has become increasingly important for retailers. While product information can sometimes be effectively and efficiently personalized to the individual level, there are common situations where this is not the case, e.g., e-mailed "mini-catalogs" and banner ads. In such cases, retailers must decide what sets of products (catalogs) should be presented to customers so as to maximize profit. While this problem falls under the framework of the single mailing catalog creation problem introduced in [KPRS98], that paper did not consider the case of multiple mailings and the algorithms which that paper introduced for solving the single mailing problem are not practical for any but the smallest problems.

In this paper we have presented algorithms that efficiently solve both the single mailing and multiple mailings problems. There are three approaches for the single mailing problem: a gradient ascent approach (DCC) that attempts to directly optimize the objective (e.g. maximize profit), a customer clustering approach (ICC) that indirectly attempts to optimize results by grouping similar customers, and a hybrid of these two approaches (HCC). The combined approach appears to be the best technique, although the gradient ascent approach outperforms it when the number of products is small. Finally, our technique for solving the multiple mailings problem finds a solution by simply and efficiently solving a series of single mailing problems.

The performance of our algorithms with respect to what is optimally possible is a key issue. However, we were able analyze the behavior of our algorithms with respect to a theoretical bound and, in some cases, show good performance for both the single mailing and multiple mailings algorithms. However, this is an area for further investigation.

7 References

[CKPT92] Douglass R. Cutting, David R. Karger, Jan O. Pedersen, and John W. Tukey, "Scatter/Gather: A Cluster-based Approach to Browsing Large Document Collections", ACM SIGIR '92, Pages 318 – 329, 1992.

[CLR90] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest, *Introduction to Algorithms*, Prentice Hall, 1990.

[DJ88] Richard C. Dubes and Anil K. Jain, *Algorithms for Clustering Data*, Prentice Hall, 1988.

[KPR98] Jon Kleinberg, Christos Papadimitriou, and Prabhakar Raghavan, "Segmentation problems," STOC '98, pages 473-481, 1998.

[Kow97] Gerald Kowalski, *Information Retrieval Systems – Theory and Implementation*, Kluwer Academic Publishers, 1997.

[KR90] L. Kaufman and P. J. Rousseeuw, *Finding Groups in Data: an Introduction to Cluster Analysis*, John Wiley and Sons, 1990.

[LA99] Bjorner Larsen and Chinatsu Aone, "Fast and Effective Text Mining Using Linear-time Document Clustering," KDD-99, San Diego, California, 1999.

[Rij79] C. J. van Rijsbergen, *Information Retrieval*, Buttersworth, London, second edition,1989.

[SKK00] Michael Steinbach, George Karypis, and Vipin Kumar, "A Comparison of Document Clustering Algorithms," KDD-2000 Text Mining Workshop, 2000.