Targeted Advertising with Inventory Management

David Maxwell Chickering and David Heckerman
Microsoft Research
Redmond WA, 98052-6399
dmax@microsoft.com
heckerma@microsoft.com

August 25, 2000

Technical Report
MSR-TR-00-49

Microsoft Research
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052

Abstract

Companies that maintain web sites can make considerable revenue through advertising, and consequently attracting advertisers has become an important and competitive endeavor. A property that can attract advertisers is high click-through rates; and therefore companies can benefit from delivery systems that serve advertisements selectively to those visitors most likely to click. In order to satisfy contractual obligations, however, these systems must simultaneously perform inventory management. For example, if a company has agreed to serve a certain number of a particular advertisement, it must do so regardless of how likely it is to be clicked. In this paper, we describe how to use a linear program to identify a schedule, based on known attributes of each visitor, that maximizes the expected number of clicks given all of the inventory-management constraints. We present experimental results using real data that demonstrate that a delivery schedule from our system realizes more clicks than a schedule that was hand constructed.

1 Introduction

Advertising revenue on the web is proving to be important for many companies that host web sites, as the resulting revenue can allow those companies to make a profit without charging visitors for using their site. See (e.g.) Hoffmann, Novak, and Chatterjee (1995) for a discussion of the business model of sponsored content sites.

As discussed by (e.g.) Baudisch and Leopold (1997), many companies have turned to *targeting* to compete for advertising dollars. The idea is to employ advertisement-delivery systems that use collected information about the visitors to decide which advertisements to show. For example, if a visitor to a news site reads many sports stories, then a delivery system can infer that the visitor is probably interested in sports and serve ads accordingly. In addition, it may be possible for a system to use explicitly collected data about the visitor such as answers to a questionnaire.

The objective of targeting from the hosting web site's point of view is to convince advertisers that the targeting is likely to lead to increased sales. There are two common targeting approaches that companies use to achieve this objective.

The first, which we call *targeted branding*, is to try to show the advertisements to a specific segment of users. The idea is to increase the brand presence of the advertiser, and consequently the advertiser's sales, among that segment of users. Consider, for example, a tennis-shorts company that wants to advertise on the web. It seems reasonable that the company would be interested in increasing brand awareness only among people who are interested in tennis. A sports-related web site using targeted branding could promise the tennis-shorts company to show a certain number of their ads on pages that contain tennis

news. Alternatively, a company with a more sophisticated targeting system may promise the advertiser to show a large percent of the advertisements to males who are interested in tennis.

In the second approach, the web site tries to maximize the number of times visitors click on the advertisements. When a visitor clicks on an advertisement, their browser is usually redirected to the advertising companies' web site where they might purchase a product.

Of the two approaches, targeted branding is the most difficult to evaluate. First, measuring brand presence is a difficult—but not impossible (see Briggs and Hollis, 1997)—task. Second, it is often difficult or impossible to know whether a particular visitor fits the desired segment that is being targeted.

In contrast, evaluating the number-of-clicks approach is straightforward. A web site can use above-average clicks as a selling point to advertisers, and back up claims with real data.

Whether targeting for branding, click-throughs, or a combination of the two, a web site will be faced with inventory-management constraints: web sites typically enter into contracts with advertisers and promise to deliver a certain number of impressions[1] of one or more advertisements. This means that both (1) a site must be careful not to over sell to advertisers, and (2) the delivery system is constrained to deliver all of the advertisements sold, regardless of the type of visitors that come to the site.

In this paper, we concentrate on serving advertisements to maximize the number (or equivalently, the overall rate) of clicks at a web site. In particular, we describe how to use a linear program to construct an advertisement-delivery system that maximizes the expected click rate, given the inventory-management constraints. In Section 2, we describe the details of our solution. In Section 3, we present experimental results demonstrating that the described system improves click rates on a real web site. In Section 4, we describe extensions to our approach. Finally, in Section 5, we summarize our work. The use of a linear program to solve a similar advertisement-delivery problem was developed independently by Langheinrich, Nakamura, Abe, Kamba and Koseki (1999). To our knowledge, our paper is the first to validate the approach in a real setting.

2 A Linear Program for Optimizing Clicks

In this section, we describe how to construct a linear program to identify an optimal delivery schedule for sites interested in maximizing clicks.

Our approach is based on assigning each visitor to a unique cluster, and then using the cluster of the visitor to decide which advertisement(s) to show. We assume that the

[1]When an advertisement is shown to a visitor of a page, we say that the site has *delivered an impression.*

total number times an advertisement is shown to a cluster is reasonably large so that the probability of clicking an advertisement given a known visitor cluster can be estimated reliably from data. A cluster can be as simple as the particular area of a web site the visitor is currently navigating; or a cluster can be determined by an arbitrary function of all the information we know about the visitor. As an example of the latter, Chickering, Heckerman, Meek, Platt and Thiesson (2000) apply machine-learning techniques to collected data to identify clusters of visitors that have similar clicking behavior among the advertisements.

Our approach consists of two phases. In the first phase, the delivery system delivers advertisements uniformly to all clusters and collects statistics about each advertisement. In particular, for each advertisement and for each cluster, the system records (1) the number of times the advertisement was shown to visitors in the given cluster, and (2) the number of times visitors in the given cluster clicked on the advertisement. Using these counts we can estimate, for each cluster, the probability that a visitor in that cluster clicks on the advertisement. The first phase need only be run long enough to get accurate probability estimates. For large web sites, we can obtain reasonable estimates in less than a day.

In the second phase of our approach, we use the estimated click probabilities to construct a new schedule that maximizes the expected number of clicks. Before explaining this phase, we need some notation. Assume there are m clusters and n advertisements. We use p_{ij} to denote the probability, estimated in the first phase of the algorithm, that advertisement i will be clicked if shown to a visitor in cluster j. A particular delivery schedule is defined by the set $\mathbf{X} = \cup_{i,j}\{x_{ij}\}$, where x_{ij} is the number of times that advertisement i is to be shown to a visitor in cluster j in a given period of time (e.g., one day).

Given this notation, and assuming that the click probabilities do not depend on the schedule, we can express the expected number of clicks for any schedule $\mathbf{X}$ as:

$$E(\text{Number Clicks}) = \sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij} \cdot x_{ij} \tag{1}$$

Let q_i denote the quota (number of promised impressions) for advertisement i, and c_j denote the size of cluster j. The units of q_i and c_j are defined for the same time period as x_{ij}. For example, if x_{ij} is the number of times to show advertisement i in cluster j per day, then q_i is the daily quota for advertisement i, and c_j is the number of visitors per day who are in cluster j. Note that the cluster sizes can be estimated with the same counts used to estimate the click probabilities.

For each advertisement i, the quota q_i imposes the following constraint on the delivery schedule:

$$\sum_{j=1}^{m} x_{ij} \geq q_i \tag{2}$$

That is, the total number of times we serve advertisement i across all clusters must be at least as many as we promised to the advertiser. Similarly, for each cluster j, the size c_j imposes the constraint:

$$\sum_{i=1}^{n} x_{ij} \leq c_j \tag{3}$$

In other words, if only c_j visitors in cluster j visit the site per day, our daily delivery schedule should not expect to serve more than c_j advertisements to these visitors.

We would like find the schedule $\mathbf{X} = \cup_{i,j}\{x_{ij}\}$ that maximizes Equation 1, subject to the inventory-management constraints expressed in Equations 2 and 3. Because the objective function (expected number of clicks) is a linear function of $\mathbf{X}$, and all the constraints are also linear functions of $\mathbf{X}$, we can identify the optimal schedule using a linear program. For a good description of linear programming, see (e.g.) Chvátal (1983).

Once the optimal schedule $\mathbf{X}$ has been identified the delivery system simply needs to deliver x_{ij} impressions of advertisement i to visitors in cluster j. A simple way to show approximately the right number of each advertisement is as follows: when a visitor in cluster j is to be served an advertisement, we randomly choose to serve advertisement i with probability:

$$\frac{x_{ij}}{\sum_{i'} x_{i'j}}$$

This approach has the advantage that the system does not need to keep track of which advertisements have already been served. Furthermore, the random nature of the algorithm ensures that any particular visitor is likely to be shown a variety of advertisements.

A potential problem with our approach as described is that the solution to the linear program can be sensitive to small errors in the p_{ij} estimates. For example, suppose that for two different clusters, the corresponding "true" click rates for a particular advertisement are identical and equal to 0.5. Even with a reasonably large sample, we are almost guaranteed to have two different estimates for the two rates. Suppose that one of the estimated rates is 0.501 and the other is 0.499. In this case, the optimal solution is likely to place all of the advertisement impressions into the cluster with the higher rate. We would prefer a more uniform placement of advertisements for two reasons. First, visitors in a given cluster will get a better variety of advertisements. Second, we expect the resulting expected number of clicks to be less sensitive to errors because a smaller percent of them will depend on

particular p_{ij} values. Extending the linear-program solution of Langhienrich et al. (1999), Tomlin (2000) independently solved this problem by optimizing a *non-linear* function of $\mathbf{X}$ that trades off the number of clicks with the uniformity of the solution.

Our approach to avoid sensitivity to the probability estimates is to bucket the probabilities. The idea is that we identify ranges of probabilities, which we call *buckets*, and then we replace each p_{ij} with the mean of the bucket into which it falls. Given a desired number of buckets k, we use the following simple algorithm to identify the buckets. Initially, we have a separate bucket for each value p_{ij}. Then, as long as we have more buckets than k, we calculate the mean value for the probabilities in each bucket, and merge together those two buckets that have the smallest difference in means. The best choice for k will depend on the domain, and should be tuned with some experimentation.

When bucketing is added to the delivery system, there can be many optimal schedules because of the ties in the bucketed click rates. Our system can be modified to find the most uniform optimal schedule when we use bucketing. The modification works as follows. First, we collect data as before, and define the p_{ij} values using bucketing. Next, we run the original linear program to identify the optimal number of clicks, denoted C. Then, we define a *second* optimization problem that identifies an optimal schedule that is "closest" to the schedule where each advertisement is placed uniformly in each cluster. In particular, we *minimize* the following objective function:

$$\sum_{i=1}^{n}\sum_{j=1}^{m}|x_{ij} - \frac{q_i}{m}| \tag{4}$$

Recall that q_i is the number of impressions to serve for advertisement i, and that m is the total number of clusters. If we place an equal number of impressions of a particular advertisement i' in each of the m clusters, we will have $x_{i'j} = \frac{q_{i'}}{m}$ for all j. Equation 4 simply measures the distance in impressions, for each advertisement, from this uniform configuration.

The constraints of the optimization problem include all of the constraints from the original problem (Equations 2 and 3) with the added constraint that the expected number of clicks in the solution is the same as the (optimal) number identified in the first linear program. In particular, given an optimal number of clicks (C) identified by the first linear program, we add the following constraint to the second optimization:

$$\sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij} \cdot x_{ij} = C$$

The secondary optimization thus identifies the most uniform delivery schedule, subject to the inventory-management constraints, and subject to the constraint that the schedule

must have the maximum expected clicks. It is well known that a linear program can be used to solve our secondary optimization; there is an easy transformation that can eliminate the absolute-value terms in Equation 4 so that we are left with a linear objective and linear constraints.

We mentioned that, in the data-collection phase of the algorithm, advertisements were to be served *uniformly* to all clusters. The idea is that we would like to have reasonable estimates for all p_{ij}. In fact, the probability p_{ij} need not be estimated if we do not plan to show advertisement i to people in cluster j. As an example, suppose a cluster corresponds to people navigating the sports area of a web site, and an advertiser makes a specific request *not* to show any advertisements in this cluster. We can implement this as a linear constraint ($x_{ij} = 0$), and then the value for p_{ij} is irrelevant to our optimization. An alternative (and more sensible) implementation is to simply remove all terms containing x_{ij} from the optimization.

New advertisements can be added dynamically to our system relatively easily as long as the current schedule has not filled the capacity of the clusters. In particular, we may be able to collect data (phase one) for a new set of advertisements, while an existing (optimal) delivery schedule is being implemented. After collecting statistics, we find a new optimal schedule that includes the new advertisements. Even easier is the deletion of advertisements from the schedule, as we simply need to re-optimize with less advertisements, using the existing p_{ij} values that are still relevant.

3 Experimental Results

In this section, we describe some experimental results from applying our approach on the MSNBC web site.

In our first experiment, we performed a *passive* test of our system on the entire site. That is, our schedule was not actually implemented by MSNBC. MSNBC.com is organized into *page groups*, where each page group corresponds to a broad class of news stories. At the time of the experiment, there were roughly 20 page groups and roughly 500 advertisements. Examples of page groups include a sports page group and a technology page group. We used the page groups to determine the clusters. In particular, when someone visited a particular page on MSNBC.com, we simply determined the corresponding page group of that page to define the visitor's cluster. Advertisements are normally scheduled on MSNBC manually; advertisers buy impressions on page groups. For example, an advertiser may choose to buy 1000 impressions for the sports group and another 1000 impressions for the technology group. Impressions sold within a particular page group are randomly served to users that

visit pages within that page group.

We collected two days worth of web logs on December 21 and December 22 of 1998. We used roughly 1.6 million advertisement impressions from the first day to estimate the click probabilities and cluster sizes. Each probability was estimated using an average of 4,000 data points. Then we ran the linear program to identify the schedule that maximized the expected number of clicks. In this experiment, we did not use bucketing, and consequently we did not perform the secondary optimization. The linear program identified the optimal schedule in less than a minute.

As mentioned, the experiment was passive; we used the data from the second day to estimate how well the resulting schedule would have worked. In particular, we used the data from the second day—which consisted of roughly the same number of page views as the first—to re-estimate all of the p_{ij} values, and then we calculated the expected number of clicks for the optimized schedule via Equation 1. We compared this number to the *actual* number of clicks seen on the second day, and found that our approach yielded an improvement of between 20 and 30 percent[2].

Given our positive results, a particular advertiser agreed to let us re-arrange their advertisements in an active experiment. The advertiser had five advertisements running across MSNBC, and was interested in how much we could improve their total click rate by re-arranging the advertisements across the site. For this experiment, we trained the parameters using two days of web logs over the weekend of May 15, 1999. Each parameter was estimated using roughly 15,000 data points. We bucketed these parameters as described in the previous section using roughly ten buckets. Then, we used our linear program to identify a schedule with maximum clicks. Next, we ran the secondary optimization, using the given maximum clicks as a constraint, to identify the most uniform optimal schedule. Because the number of advertisements was small, both linear programs completed in under a second. Finally, we implemented this schedule during the following weekend. The results of this experiment are shown in Figure 1.

In the figure, we have included the click rate of all of the other advertisements on both weekends[3]. The click rate for these advertisements did not change significantly between the two weekends. In contrast, our approach yielded a 30 percent increase in click rate for the advertisements we re-scheduled. We note that the predicted increase (the expected number of clicks in the optimized schedule given the estimates of p_{ij} from the training data versus the actual number of clicks generated by the original schedule) was also 30 percent.

[2]The variability comes from different smoothing methods for estimating the parameter values p_{ij}.

[3]We have deliberately omitted the absolute magnitudes of the click rates, and compare the relative improvement of our approach.

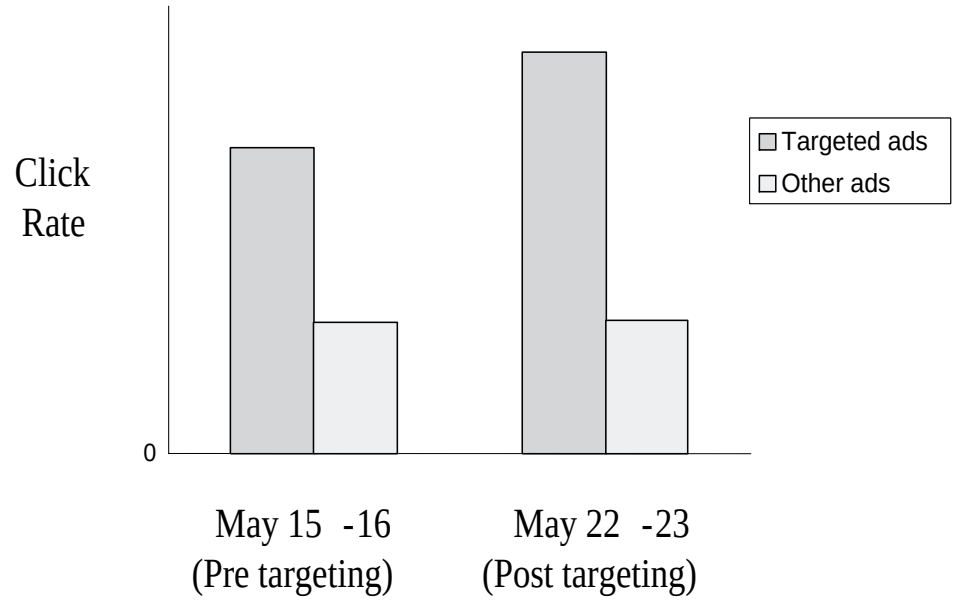

Figure 1: Results of an active implementation of our system on MSNBC.

4 Extensions

Our approach can be used to optimize *any* linear function of $\mathbf{X}$, not just the total expected clicks. As an example, we could add a constant α_{ij} to each term in Equation 1 that weights the importance of showing the given advertisement. This allows the site to give preferential treatment to (e.g.) advertisers who pay more.

The ability to change the objective function in our system addresses a possible objection to our approach: advertisers are not really interested in clicks, but rather they are interested in increasing profits. Assuming the data is available, it is easy to construct an appropriate (linear) objective function to maximize. For example, if each p_{ij} term from Equation 1 is re-defined to denote the probability that a user in cluster j will make a *purchase* corresponding to advertisement i, the system can be applied directly to find the schedule that maximizes the number of purchases. As another example, suppose that for each cluster j we can estimate the expected profit r_{ij} (e.g. in dollars) that will result from showing each advertisement. Then our system can maximize the total expected revenue for all advertisers ($\sum_{ij} r_{ij} \cdot x_{ij}$), using the same inventory-management constraints from the original formulation of the problem.

The schedule that maximizes the total number of clicks for all advertisers may drasti-

cally reduce the number of clicks for a particular advertiser. In another extension, we can explicitly prevent this from happening (in expectation) by adding the constraint that the total number of expected clicks for each particular advertiser must be at least as large as in the pre-targeted schedule. As another example, we can implement targeted-branding solutions into our system by allowing advertisers to insist that a certain number of advertisement impressions remain in particular clusters, while allowing the remaining impressions to be optimized for clicks.

5 Summary

In this paper, we have described a system that, using a linear program, identifies an advertisement-delivery schedule that maximizes the number of clicks while maintaining the inventory-management constraints of a web site. We described an extension to the system that uses bucketing and a second linear program to find the most uniform such schedule. We have demonstrated in both a passive and an active experiment that the method is effective.

References

Baudisch, P. and Leopold, D. (1997). User-configurable advertising profiles applied to web page banners. In *Proceedings of the First Berlin Internet Economics Workshop*, Berlin.

Briggs, R. and Hollis, N. (1997). Advertising on the web: Is there response before click-through? *Journal of Advertising Research*, 37(2):33–45.

Chickering, D. M., Heckerman, D., Meek, C., Platt, J. C., and Thiesson, B. (2000). Goal-oriented clustering. Technical report, MSR-TR-2000-82, Microsoft, Redmond, WA.

Chvátal, V. (1983). *Linear Programming.* W. H. Freeman and Company, New York.

Hoffman, D., Novak, T., and Chatterjee, P. (1995). Commercial scenarios for the web: Opportunities and challenges. *Journal of Computer-Mediated Communication,* special issue on Electronic Commerce, 1(3).

J.A.Tomlin (2000). An entropy approach to unintrusive targeted advertising on the web. In *Proceedings of WWW9.*

Langheinrich, M., Nakamura, A., Abe, N., Kamba, T., and Koseki, Y. (1999). Unintrusive customization techniques for web advertising. In *Proceedings of WWW8.*