



Improvements to the Linear Programming Based Scheduling of Web Advertisements*

ATSUYOSHI NAKAMURA

atsu@main.eng.hokudai.ac.jp

Division of Systems and Information Engineering, Graduate School of Engineering, Hokkaido University, Sapporo 060-8628, Japan

NAOKI ABE

nabe@us.ibm.com

IBM Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598, USA

Abstract

We propose and evaluate a number of improvements to the linear programming formulation of web advertisement scheduling, which we have proposed elsewhere together with our colleagues [Langheinrich et al., 9]. In particular, we address a couple of important technical challenges having to do with the estimation of click-through rates and optimization of display probabilities (the exploration–exploitation trade-off and the issue of data sparseness and scalability), as well as practical aspects that are essential for successful deployment of this approach (the issues of *multi-impressions* and inventory management). We propose solutions to each of these issues, and assess their effectiveness by running large-scale simulation experiments.

Keywords: banner advertisement scheduling, linear programming, exploration–exploitation trade-off, inventory management

Banner advertisement on the Internet is a new form of advertisement which, by allowing the users to click on the banner of their choice and obtain information and conveniences, provides a type of interactive service that was not possible with any conventional form of advertisement. With the Internet and electronic commerce becoming widely used, Internet advertisement is attracting attention as an important source of both information and revenue. As the importance and volume of Internet advertisement (or simply Internet ad) industry grow, the need for an intelligent and adaptive scheduling method for Internet ads has become acute, which, based on the past behavior of users, learns to display ads that will yield the highest possible click-through rate. Such a scheduling method would benefit both the advertisers and users, as it allows the advertisers to direct ads to more targeted populations, and the users to enjoy a greater convenience. It should also benefit the Internet providers and advertising agents by helping them increase their ad revenues. Pricing of banner ads is increasingly based on the number of click-throughs, rather than the number of impressions, and higher click-through rates should translate to higher prices.

* Most of the work presented herein was performed while these authors were with NEC Corporation. Parts of the contents of this paper have appeared in [Abe and Nakamura, 1; Nakamura, 11].

There have been numerous customization and personalization techniques developed in the areas of data mining and information filtering, which, at first glance, might appear to be good candidates for techniques of intelligent ad targeting and scheduling. These include customer segmentation using classifiers such as decision trees [Quinlan, 14], and recommendation engines using collaborative and content-based filtering [Pazzani, 13; Nakamura and Abe, 12]. These methods, however, suffer from at least two major drawbacks as methods of ad targeting and scheduling. First, many of them require collection of various information about the users, demographic, transactional or otherwise, which is technically infeasible at non-member web sites, as well as undesirable from the viewpoint of privacy (cf. [Langheinrich et al., 9]). Second, these methods do not deal with constraints inherent in ad targeting, such as the number of promised impressions in advertisement contracts. For this reason, we have proposed, together with our collaborators, a novel approach to ad targeting based on linear programming (LP) that achieves high click-through rates while adhering to various constraints [Langheinrich et al., 9]. Subsequently, other authors have expanded upon this approach: Tomlin [17] and Chickering and Heckerman [4] pointed out the “over-targeted” feature of the LP approach, and proposed modified LP models that incorporate some forms of randomization. Notably Chickering and Heckerman [4] have demonstrated the effectiveness of the LP approach by conducting real world experiments on a web site open to the public.

In the present paper, we address a number of additional issues that must be dealt with, in order to realize a practical ad targeting solution based on the LP approach. The first two issues we study are technical aspects having to do with estimation and optimization. The first one is the exploration–exploitation trade-off, namely that of resolving the trade-off between displaying ads that help the system learn about the users behavior versus those that bring more immediate rewards (clicks). We propose a number of modifications to the original linear programming formulation of Langheinrich et al., each of which deals with this issue. These include the use of Gittins Index in place of the estimated click through rates in the objective function, the use of an interior point method for linear programming as a semi on-line learning method to allow gradual convergence, and a lower bounding technique which designates a certain portion of the display probabilities to all ads. Another important issue is how to cope with a potentially large cardinality of the attribute space, which poses a challenge both in terms of the data sparseness for the estimation of click rates and the high dimensionality of the underlying LP problem. To address the latter issue, we devise an efficient clustering algorithm that clusters attributes having similar click through statistics. This method allows us to improve the accuracy of click through rate estimates, as well as to reduce the dimensionality of the LP problem.

We also address some practical aspects that are important for deployment of this approach. One is how to deal with *multi-impressions*, in which two or more banner ads are displayed at the same time. The difficulty here is that if we independently select each of the multiple advertisements that are to be displayed on the same web page, then the same ad could appear more than once. For this issue, we augment the probabilistic selection mechanism with a queue holding ads that have been selected but cannot be immediately displayed due to a conflict. The other aspect we address is the issue of inventory management. Over-selling can cause the linear programming formulation of the scheduling problem to have

no solution. This problem is challenging since multiple advertisers can make requests involving related constraints, and thus inventory questions are inter-dependent. We in fact solve this latter issue by formulating the problem as another linear programming problem.

We ran large-scale simulation experiments in order to assess the effectiveness of some of the proposed methods. The results of our experiments indicate the following, modulo the assumption that the simulation model used in the experiments reflects the relevant characteristics of the real world.

- (1) Various measures for the exploration–exploitation trade-off improve the basic method based on linear programming in terms of the total click rate achieved. Relative merits and demerits of alternative measures were found to be dependent on the learning interval and the campaign length.
- (2) Clustering of the attributes improves the total click through rate.
- (3) The computational requirement of our method is small enough to render it practical.
- (4) Multi-impressions can be handled using relatively short queues by reducing the display probabilities by reasonably small margins.

An important future problem is to verify, via real world experiments, whether these observations hold in actual applications.

1. The linear programming model

In this section, we review the Linear Programming (LP) model of banner advertisement scheduling, as proposed by Langheinrich et al. [9]. In the original paper, Langheinrich et al. primarily considered keyword-driven and page-driven ads. Clearly, it is possible to have attribute-driven ads for other attributes, and more generally, “attribute-set-driven ads” for any attribute set. Here, we assume that there is a fixed set of attributes we are interested in, such as the set consisting of ‘time of day’ and ‘page category.’

Before describing the formalization, we first show an example which helps illustrate the problem and the need for an LP solution. Assume that the accurately estimated numbers of page views for combinations of attribute values (afternoon, sports), (afternoon, not sports), (not afternoon, sports) and (not afternoon, not sports) are 10,000, 10,000, 5,000 and 5,000, respectively. Also assume that there are three ads for each of which 10,000 impressions have been promised, and that the accurately estimated click-through rates of these ads for the combinations of attribute values are as shown in Table 1.

Table 1. Case that the local strategy fails.

Time of day	Page category	Number of page views	Click-through rate (%)		
			ad 1	ad 2	ad 3
afternoon	sports	10,000	2.2	1.1	1.0
afternoon	not sports	10,000	2.2	2.1	1.0
not afternoon	sports	5,000	2.2	2.1	2.0
not afternoon	not sports	5,000	2.2	2.1	2.0

Consider the greedy strategy that selects an ad with the highest click-through rate among the ads whose promised number of impressions has not been achieved yet. We can easily see that this strategy, which seems to be often employed in practice, does not work in the above case. Assume that page views for all combinations of attribute values occur randomly.¹ The greedy strategy always selects ad 1 for the first 10,000 page views, ad 2 for the second 10,000 page views and ad 3 for the last 10,000 page views, because (the click-through rate of ad 1) > (the click-through rate of ad 2) > (the click-through rate of ad 3) holds for all combinations of attribute values. As a result, we find that the actual click-through rates for ad 1, ad 2 and ad 3 are 2.2%, 1.76...% and 1.33...%, and the total click-through rate for all ads is 1.76...%, which is the same rate as what would be obtained by the random selection strategy.

In the above case, according to the optimal display schedule in the LP model, ad 1 is always selected for (afternoon, sports), ad 2 for (afternoon, not sports) and ad 3, otherwise. The total click-through rate of this optimal schedule is 2.1% and the actual click-through rates for ad 1, ad 2 and ad 3 are 2.2%, 2.1% and 2.0%, respectively.

Now let us describe the problem formulation. Let i ($= 1, \dots, n$) denote attribute value combinations and j ($= 1, \dots, m$) denote ads. We define k_i , h_j and $c_{i,j}$ as follows:

k_i : estimated rate of page views for combination i of attribute values (normalized over all combinations),

h_j : desired display rate for ad j (normalized over all ads),

$c_{i,j}$: estimated click-through probability for ad j and combination i of attribute values.

The estimation of k_i is calculated based on the number of recent page views for combination i of attribute values. Desired display rate h_j is calculated from the contracted number of impressions by subtracting the number of actual impressions so far and considering the remaining contracted period. The estimation of $c_{i,j}$ is calculated based on the recent click-through rate for combination i of attribute values and ad j .

Let $d_{i,j}$ denote the display probability of ad j for combination i of attribute values. The problem we have to solve is the following problem (Problem 1).

Problem 1. Maximize

$$\sum_{i=1}^n \sum_{j=1}^m c_{i,j} k_i d_{i,j} \quad (1)$$

under the conditions

$$\sum_{i=1}^n k_i d_{i,j} = h_j \quad \text{for } j = 1, \dots, m, \quad (2)$$

$$\sum_{j=1}^m d_{i,j} = 1 \quad \text{for } i = 1, \dots, n, \quad (3)$$

$$d_{i,j} \geq 0 \quad \text{for } i = 1, \dots, n, \quad j = 1, \dots, m. \quad (4)$$

Problem 1 is an instance of the “transportation problem” (see, e.g., [Dantzig, 5]) and efficient algorithms (see, e.g., [Bradley et al., 2]) are known. With this model, constraints specified by advertisers can be naturally incorporated. Assume that ad j_0 is *not* allowed to be displayed for combination i_0 of attribute values by a certain contract with an advertiser. Then, one can incorporate this constraint into Problem 1 by considering pair $(i, j) \neq (i_0, j_0)$ only. Another approach is to set c_{i_0, j_0} to -1 , which prevents the problem from having no solution (see [Langheinrich et al., 9]).

There is a practical issue with this original formulation, having to do with the fact that the optimal solution of Problem 1 maximizes *total* click-through rate for *all* ads. This means in particular that some ads may be assigned to areas (attribute combinations) with low click-through rate, in order to obtain a high overall click-through rate. Consider an example, wherein 10,000 impressions each have been promised for two ads. Assume that the accurately estimated click-through rates (%) of these ads for combination 1 and 2 of attribute values are as shown in the following table.

	ad 1	ad 2
Combination 1 of attribute values	4.0	2.5
Combination 2 of attribute values	2.0	1.0

Then, the optimal solution² is the one that always displays ad 1 for combination 1 and ad 2 for combination 2. With this solution, ad 1 will have a high click-through rate of 4.0% while ad 2 suffers from having a low click-through rate of 1.0. This can be a problem, for example, if the advertiser of ad 2 is more important than the advertiser of ad 1. This problem can be dealt with, to some extent, by introducing the ‘degree of importance’ $g_j > 0$ for each ad j . The default value of g_j is 1.0, and a greater value indicates a greater importance being assigned to ad j . We modify the objective function (1), in the linear programming formulation, by the following modification, which incorporates the degrees of importance:

$$\sum_{i=1}^n \sum_{j=1}^m g_j c_{i,j} k_i d_{i,j}. \quad (5)$$

If degrees of importance for ad 1 and ad 2 are 1.0 and 2.0, respectively, the optimal solution for the above example is the one that always displays ad 2 for combination 1 and always displays ad 1 for combination 2. This decreases the click-through rate of ad 1 from 4.0 to 2.0, but increases that of ad 2, which is presumably more important than ad 1, from 1.0 to 2.5. In practical operations, if one finds that a certain ad of importance has a low click-through rate after a portion of its contract period has passed, one can improve the click-through rate by increasing its degree of importance.

2. Data sparseness and scalability

2.1. Estimating the click rates via clustering

In the setting of keyword directed ad scheduling, the number of keywords is normally in the thousands or hundreds, even if we collect the infrequently input ones together as ‘all

others.’ The situation only worsens when one considers attribute set driven ads, since the number of attribute value combinations is exponential. Thus, especially near the beginning of a campaign period when the data are sparse, the click rate estimates are highly unreliable and the solution of the above linear programming problem may be led astray. As a remedy for this problem, we formulate the click rate estimation problem as the estimation of a conditional probability model $\theta_{\mathcal{P}} = \{P(\text{click}|Z, j): Z \in \mathcal{P}, j = 1, \dots, m\}$ of the form:

$$P(\text{click}|i, j) = P(\text{click}|Z, j), \quad i \in Z,$$

where the \mathcal{P} is a partition of attribute value combinations ($Z_1 \cap Z_2 = \emptyset$ for all $Z_1, Z_2 \in \mathcal{P}$ with $Z_1 \neq Z_2$, $\bigcup_{Z \in \mathcal{P}} Z = \{1, \dots, n\}$). This problem includes model estimation, that is, the estimation of a partition \mathcal{P} or clustering of attribute value combinations. Since the aforementioned linear programming problem has dimension mn , where m is the number of ads and n is the number of attribute value combinations, clustering should contribute greatly to the reduction of dimensionality and hence the computational complexity, in addition to the improvement of estimation accuracy.

For estimating the above model, we devised an efficient estimation method based on an information criterion known as AIC (Akaike’s Information Criterion) [Sakamoto et al., 16]. We also tried using MDL (Minimum Description Length principle) [Rissanen, 15], but it turned out that AIC worked better for our purpose. Results of experiments comparing the two methods will be described in Section 6.3. Our clustering method stipulates that the probability model minimizing $I(\theta_{\mathcal{P}}) = LL(\theta_{\mathcal{P}}) + PT(\theta_{\mathcal{P}})$, where $LL(\theta_{\mathcal{P}})$ is the minus log likelihood of the model and $PT(\theta_{\mathcal{P}})$ is the penalty term representing the number of free parameters in the model. The minus log likelihood of a model, as its name implies, is the minus logarithm of the likelihood (probability) of a model given the data. So the minus log likelihood for $\theta_{\mathcal{P}}$ is

$$LL(\theta_{\mathcal{P}}) = \sum_{Z \in \mathcal{P}} \sum_{j=1}^m -\left(C_{Z,j} \log P(\text{click}|Z, j) + (D_{Z,j} - C_{Z,j}) \log(1 - P(\text{click}|Z, j)) \right),$$

where $D_{Z,j}$ is the number of displays (or impressions) for the cluster ad pair Z, j and $C_{Z,j}$ is the number of clicks observed for the same pair. It is well known and straightforward to verify that this is minimized by letting $P(\text{click}|Z, j) = C_{Z,j}/D_{Z,j}$, so the *minimum* minus log likelihood for a given partition \mathcal{P} is given as follows.

$$LL(\theta_{\mathcal{P}}) = \sum_{Z \in \mathcal{P}} \sum_{j=1}^m -\left(C_{Z,j} \log \frac{C_{Z,j}}{D_{Z,j}} + (D_{Z,j} - C_{Z,j}) \log \frac{D_{Z,j} - C_{Z,j}}{D_{Z,j}} \right).$$

The penalty term, according to AIC, is the number of free (probability) parameters in a model, and is simply

$$PT(\mathcal{P}) = m|\mathcal{P}|.$$

We let $I(\mathcal{P})$ denote $I(\{C_{Z,j}/D_{Z,j}: Z \in \mathcal{P}, j = 1, \dots, m\})$. Now, the minimization of $I(\theta_{\mathcal{P}})$ is reduced to that of $I(\mathcal{P})$.

Algorithm CLUSTER+
Input: $\{(D_{i,j}, C_{i,j}): i = 1, \dots, n, j = 1, \dots, m\}$
Output: A partition \mathcal{P} of $\{1, \dots, n\}$
 $\mathcal{P} = \{\{1\}, \dots, \{n\}\}$
For all $(i, j) \in [n] \times [m]$
 $D_{(i),j} = D_{i,j}, C_{(i),j} = C_{i,j}$
For all $Z_1 \in \mathcal{P}$
 $\text{mdp}(Z_1) = \arg \max_{Z_2 \in \mathcal{P}} (I(\{Z_1, Z_2\}) - I(\{Z_1 \cup Z_2\}))$
 $\text{mda}(Z_1) = I(\{Z_1, \text{mdp}(Z_1)\}) - I(\{Z_1 \cup \text{mdp}(Z_1)\})$
Repeat
 $Z_1^* = \arg \max_{Z_1 \in \mathcal{P}} \text{mda}(Z_1)$
If $\text{mda}(Z_1^*) \leq 0$ return \mathcal{P}
 $Z_2^* = \text{mdp}(Z_1^*)$
 $\mathcal{P} = \mathcal{P} \cup \{Z_1^* \cup Z_2^*\} - \{Z_1^*, Z_2^*\}$
For $j = 1$ to m
 $D_{Z_1^* \cup Z_2^*, j} = D_{Z_1^*, j} + D_{Z_2^*, j}, C_{Z_1^* \cup Z_2^*, j} = C_{Z_1^*, j} + C_{Z_2^*, j}$
For all $Z_1 \in \mathcal{P}$
If $(Z_1 = Z_1^* \cup Z_2^*)$ or $(\text{mdp}(Z_1) = Z_1^* \text{ or } Z_2^*)$ then
 $\text{mdp}(Z_1) = \arg \max_{Z_2 \in \mathcal{P}} (I(\{Z_1, Z_2\}) - I(\{Z_1 \cup Z_2\}))$
Else
 $\text{mdp}(Z_1) = \arg \max_{Z_2 \in \{\text{mdp}(Z_1), Z_1^* \cup Z_2^*\}} (I(\{Z_1, Z_2\}) - I(\{Z_1 \cup Z_2\}))$
 $\text{mda}(Z_1) = I(\{Z_1, \text{mdp}(Z_1)\}) - I(\{Z_1 \cup \text{mdp}(Z_1)\})$
End Repeat

Figure 1. An efficient clustering algorithm based on a greedy heuristic.

Since exact minimization is not computationally feasible, we have devised an efficient algorithm which we call CLUSTER+, shown in Figure 1, which efficiently finds a reasonably good model based on a greedy heuristic.³ This algorithm starts with the finest partition in which each attribute value combination constitutes its own cluster, and keeps merging the pair of combinations which reduces the value of the objective function I by the greatest amount. Doing this straightforwardly would involve calculating the decrease in I that would result by merging each of $O(n^2)$ pairs at each iteration. CLUSTER+ avoids doing so by memorizing, for each cluster Z , that cluster $\text{mdp}(Z)$ which would yield the greatest decrease in I if merged with Z , and the amount by which I decreases, $\text{mda}(Z)$, and updating them appropriately at each iteration. In particular, at each iteration, or when a new merge takes place, CLUSTER+ updates its information in terms of $\text{mdp}(Z)$ and $\text{mda}(Z)$ as follows: if $\text{mdp}(Z)$ has been merged then it need be updated by calculating the decrease in I for all other current clusters, but for all other clusters, their values need only be compared with the decrease in I that would result from merging with the new cluster.

3. Dealing with the exploration–exploitation trade-off

3.1. Using Gittins Index in the objective function

As a measure to address the issue of exploration–exploitation trade-off, we import the idea of Gittins Index [Gittins, 8] from the Bayesian theory of Bandit problems [Berry and

Fristedt, 3], and use them in place of the click rate estimates in the objective function to be maximized in the linear programming formulation. The bandit problem consists of a series of choices out of a given set of ‘arms’ of a bandit, each having a fixed but unknown success probability, and the goal is to maximize the total expected number of successes (or rewards), normally with the successes obtained far in the future geometrically discounted. (More precisely, the total discounted reward is written as $\sum_{i=1}^{\infty} \gamma^{i-1} t_i$, where $t_i = 1$ if the i th trial is a success and 0, otherwise, and γ is a constant between 0 and 1.)

It is known (by a theorem due to Gittins and Jones) that the strategy of always picking an arm with the maximum ‘Gittins Index’ is Bayes optimal, in the sense that it maximizes the expected discounted rewards. Gittins Index is defined for the statistics associated with each arm, a pair (α, β) consisting of the number of successes and number of failures for that arm, and tends to take a larger value for arms with a smaller number of trials, if the empirical success probabilities are the same. In this way, maximizing the Gittins Index is designed to automatically resolve the exploration–exploitation trade-off.

Gittins Index for an arm with associated statistics (α, β) , written $G(\alpha, \beta)$, equals the probability p such that if there were another arm with *known* success probability p then taking either arm (and doing everything the optimal way after that) would lead to the same expected total rewards. If we let $R(\alpha, \beta, p)$ denote the total expected rewards under this situation with one arm having statistics (α, β) and the other having known success probability p , then we have

$$R(\alpha, \beta, p) = \max \left(\frac{p}{1-\gamma}, \frac{\alpha}{\alpha+\beta} (1 + \gamma R(\alpha+1, \beta, p)) + \frac{\beta}{\alpha+\beta} \gamma R(\alpha, \beta+1, p) \right).$$

Thus Gittins Index $G(\alpha, \beta)$ equals that p which satisfies

$$\frac{p}{1-\gamma} = \frac{\alpha}{\alpha+\beta} (1 + \gamma R(\alpha+1, \beta, p)) + \frac{\beta}{\alpha+\beta} \gamma R(\alpha, \beta+1, p). \quad (6)$$

Such a value can be approximately calculated using dynamic programming with a reasonable amount of computation time, by approximating the indices for sufficiently large values of α and β by the corresponding success probability (click rate in our case) estimates. In practical applications it is usual to pre-calculate and store the values of indices for small enough values of α and β . Here we use Gittins Index to modify the target function being maximized in the earlier linear programming formulation, i.e., maximize

$$\sum_{i=1}^n \sum_{j=1}^m G(C_{i,j}, D_{i,j} - C_{i,j}) k_i d_{i,j}$$

instead of $\sum_{i=1}^n \sum_{j=1}^m c_{i,j} k_i d_{i,j}$ (in the formulation without degrees of importance).

3.2. *Semi on-line learning using an interior point method*

As another method to deal with the exploration–exploitation trade-off, we propose to make use of an interior point method (the affine scaling method in particular) for linear programming in a semi on-line fashion. That is, rather than fully optimizing at each learning interval, we use it to increment the previous solution by applying a small number of iteration steps at a time. The idea is that by converging more slowly to the optimum feasible solution, we ensure that enough exploration is done at early stages. The affine scaling method is an interior point method which is simpler than Karmarker’s method but is known to work efficiently in practice. It works by applying at each iteration a linear (affine) transformation (represented by matrix $D^{(k)}$) to its current solution to move it to a constant vector to ensure that the next point will be an interior point (and similarly modifying the objective function and constraints by the same transformation), and then moving along the gradient of the objective function, scaled by a constant α . (More precisely, $x^{(k+1)} = x^{(k)} + \alpha D^{(k)} d^{(k)}$, where $d^{(k)}$ is the gradient normalized to have size 1.)

Here note that we use the affine scaling method to first find a feasible solution by minimizing a modified objective function $G(x) = MC(x) - F(x)$ for some large constant M where $C(x)$ attains minimum (zero) if and only if x is feasible, and $F(x)$ is the true objective function to be maximized. This way, the feasible solution from the previous learning iteration, which is not necessarily a feasible solution with respect to the current updated statistics, can be used as an initial point to speed up the minimization of $G(x)$, as it is likely to be *nearly* feasible. We refer the interested reader to [Ye, 18], for example, for the details of this method. In our experiments, we set α to be 0.9 and applied 10 steps at each learning iteration.

3.3. *Lower bounding*

Another, perhaps more straightforward, method to address the issue of the exploration–exploitation trade-off is to bound the display probabilities from below within the linear programming formulation, so that no advertisement will receive zero display probabilities (cf. [Langheinrich et al., 9]). In particular, we add the following constraint:

$$d_{i,j} \geq \frac{1}{2m\sqrt{D_{i,j} + 1}}, \quad (7)$$

for each pair of ad j and attribute value combination i , where $D_{i,j}$ is the number of times j has been displayed on i . The rationale behind this bound is that the display probability for each pair is lower bounded by an amount that is proportional to the estimation error of the corresponding click rate. Note that the click rate estimation, for each ad-attribute value combination pair, reduces to the estimation of a bias p of a coin. Noting that the standard deviation of the frequency of heads of such a coin is $\sqrt{Np(1-p)}$, we see that the estimation error is proportional to $1/\sqrt{N}$, where N is the sample size.

4. Multi-impressions

In this section, we describe our method of realizing multi-impressions while maintaining the scheduled display probabilities. An important requirement for multi-impressions is that the multiple ads displayed on a page must be distinct.

Assume that we wish to select two different ads according to the following display probabilities.

ad 1	ad 2	ad 3
0.5	0.4	0.1

Consider the method of selecting ads according to the probability distribution restricted to those ads that have not been selected yet. In the above example, the first one is selected from the three ads according to the probabilities above, and if the first one is ad 1, then the second one is selected from the set of ad 2 and ad 3 according to probability 0.8 and 0.2, respectively, which are calculated from the above probabilities by restricting to ad 2 and ad 3 and normalizing. It can be easily verified that, when two ads are selected using this method, the display proportion of ad 1, ad 2 and ad 3 is 20 : 19 : 6, which is different from the desired proportion 5 : 4 : 1. Thus, we see that this method does not achieve the desired effect.

In our proposed method, we append those ads that are selected but cannot be immediately displayed due to conflicts, to queues from which we select at a later time. A detailed description of our selection algorithm (function) ADSelect is given in Figure 2.

Function ADSelect uses one queue for each combination of attribute values. Given a requested number k of ads and the combination i of attribute values, ADSelect returns a set S of k different ads according to the scheduled probability distribution for combination i . In doing so, it first selects as many different ads from the queue as possible ($\text{pop}(i)$), and then selects according to the scheduled probability distribution ($\text{select}(i)$). At this stage, if a selected ad f has already been selected ($f \in S$), then ADSelect appends it to the queue ($\text{append}(i, f)$). The algorithm repeats this process until the set S of selected ads contains the requested number of ads.

There is one thing that must be borne in mind, however. That is, we must set the scheduled display probabilities so as not to exceed $1/k$, where k is the number of ads displayed on the page at the same time. If this is not observed, the likelihood of queue overflow becomes very high. In fact, further consideration should convince the reader that the bound of $1/k$ is not good enough for short queue length l . To see this, notice that the overflow probability in the case with k ads having display probability $1/k$ is equal to the probability of absorption at barrier $l + 1$ in a random walk with equal “right” and “left” probabilities when $k = 2$, and is larger when $k \geq 3$. But these absorption probabilities are known to be quite high [Feller, 7].

Consider the case of $k = 2$. Assume that the queue for a certain combination i of attribute values is non-empty. Notice that all ads in the queue must be identical in this case. Let p denote the scheduled display probability of the ads in the queue. Then in the next selection of two ads, the first ad is selected from the queue, and the next one is selected randomly through the probability table. The selection of the first ad decreases

Function ADSelect*Input:* k : requested number of ads. i : combination of attribute values.*Output:* S : set of k ads.*Functions for queue operations:* $\text{pop}(i)$: the first ad in the queue for combination i of attribute values (the first ad is removed from the queue after each call of this function). $\text{append}(i, f)$: append ad f to the queue for combination i of attribute values. $\text{size}(i)$: the number of ads in the queue for combination i of attribute values.*Function for an operation concerning scheduled display probabilities:* $\text{select}(i)$: one ad selected randomly according to the scheduled display probabilities for combination i of attribute values. $M = \text{size}(i)$ $e = 0$ $S = \emptyset$

do {

 if ($e < M$) then $f = \text{pop}(i)$ else $f = \text{select}(i)$ if $f \in S$ then $\text{append}(i, f)$ else $S = S \cup \{f\}$ $e = e + 1$ } while the number of elements in S is less than k return S *Figure 2.* Ad selection algorithm for multi-impressions.

the queue length by one. The selection of the second ad might increase the queue length by more than one. The expected number of trials in the second selection is $1/(1 - p)$, and the probability of increasing the queue length by one is p at each trial. Thus, we can approximate the “increase probability” (i.e., probability that the queue length increases) by the expected number of queue pushes divided by the expected number of trials, i.e.,

$$\frac{p/(1 - p)}{1/(1 - p) + 1} = \frac{p}{2 - p}.$$

Similarly, we can approximate the “decrease probability” (probability that the queue length decreases) and the “stay probability” (probability that the queue length stays the same) by the following:

$$\frac{1}{1/(1 - p) + 1} = \frac{1 - p}{2 - p}.$$

In summary, the change in queue length is approximated by a random walk with the following probabilities:

$$\begin{aligned} \text{Increase probability: } & \frac{p}{2-p}, \\ \text{Decrease probability: } & \frac{1-p}{2-p}, \\ \text{Stay probability: } & \frac{1-p}{2-p}. \end{aligned}$$

When the queue is empty and if the display probability of the first selected ad is p , the decrease probability is 0 while the stay probability is $2(1-p)/(2-p)$. In terms of random walk, this means that there is a reflecting barrier at -1 . Assuming an absorbing barrier at $l+1$, let us consider the expected duration of this random walk. Let D_z denote the expected duration at z , that is, the expected number of steps from starting at z to absorbing at $l+1$. Then,

$$\begin{aligned} D_z &= p_{\text{inc}}D_{z+1} + p_{\text{dec}}D_{z-1} + p_{\text{sta}}D_z + 1 \quad \text{for } 0 < z \leq l, \\ D_0 &= p_{\text{inc}}D_1 + (p_{\text{dec}} + p_{\text{sta}})D_0 + 1, \\ D_{l+1} &= 0 \end{aligned}$$

hold, where p_{inc} , p_{dec} and p_{sta} are increase, decrease and stay probability, respectively. By the calculation similar to that in the case of two absorbing barriers [Feller, 7], we can obtain

$$D_0 = \frac{p(2-p)}{(1-2p)^2} \left(\left(\frac{1-p}{p} \right)^{l+1} - 1 \right) - \frac{2-p}{1-2p}(l+1).$$

In order to prevent the queue of capacity length l from overflow in N multi-impressions, this D_0 must be large enough as compared to $(1/(1-p) + 1)N$.

When $k \geq 3$, the process becomes more complicated. In particular, the increase, decrease and stay probabilities of queue length depend on how many distinct ads are in the queue. When the number of distinct ads in the queue is i and the display probability of every ad in the queue is p , these probabilities become as follows:

$$\begin{aligned} \text{increase probability: } & \frac{\sum_{j=i}^{k-1} (jp/(1-jp))}{L}, \\ \text{decrease probability: } & \frac{i}{L}, \\ \text{stay probability: } & \frac{k-i}{L}, \end{aligned}$$

where $L = \sum_{j=i}^{k-1} (1/(1-jp)) + i$. The analysis of this process is challenging because the number of distinct ads in the queue fluctuates, and we have not been able to analytically derive bounds for these cases. (We derive some empirical bounds in the section on experiments.)

The constraint on the display probabilities can be formulated as follows using the notation in the previous section:

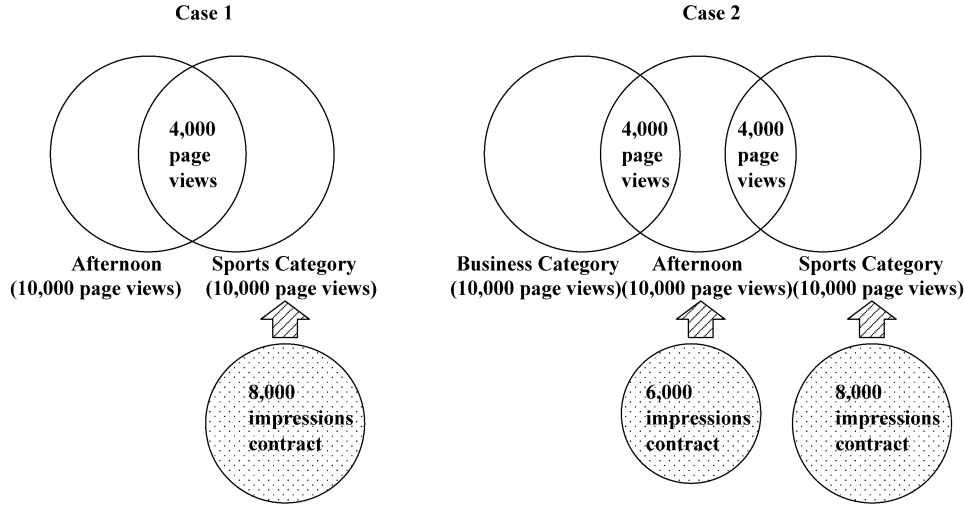


Figure 3. Two example cases of inventory status.

$$d_{i,j} \leq p_k \quad \text{for } i = 1, \dots, n \text{ and } j = 1, \dots, m, \tag{8}$$

where p_k is a display probability upper bound for k ads. Note that such constraints are actually desirable in some sense, since they help reduce one problematic feature of the LP approach that the obtained solutions tend to “over-target”, namely they tend to assign display probabilities that are too extreme. (This problem was pointed out and addressed by Tomlin [Tomlin, 17].) Since Problem 1 is still an instance of the transportation problem even with these constraints are incorporated, the modification of Problem 1 with these constraints remains efficiently solvable.

5. Inventory management

Web ad agencies and companies that host web sites generally wish to contract for as many ad impressions as possible to earn as much revenue as possible. However, over-selling results in additional cost and reduced revenue. The important thing for preventing over-selling and maximizing revenue is to grasp how many sellable impressions remain. Assuming that the estimated number of page views is accurate,⁴ this is easy if no constraints are specified by advertisers. However, this becomes non-trivial when such constraints are to be taken into account.

Consider Case 1 shown in Figure 3, where the estimated numbers of page views for afternoon and sports-category constraints are 10,000 each and 4,000 of them satisfy both constraints. Assume that there exists a contract which requests 8,000 impressions on sports-category pages only. In this case, 8,000 sellable impressions remain for afternoon constraint when no other contracts exist, since 2,000 views in (afternoon, sports) are needed for the contract for sports-category constraint. In this manner, calculation of the remain-

ing sellable impressions for a certain constraint t needs to take into account contracts for other constraints which overlap constraint t . Should only overlapping constraints be considered? Consider Case 2 in Figure 3, where the estimated number of page views is 10,000 for each of three constraints and the afternoon constraint overlaps the sports-category and business-category constraints while sports-category and business-category constraints do not overlap. The estimated page views for the two overlapping regions are 4,000 each. Assume that there exist two contracts, one requesting 8,000 impressions on sports-category pages only and the other requesting 6,000 impressions in the afternoon only. In this case, 8,000 sellable impressions for business-category constraint remain assuming no other contracts exist, and *not* 10,000 as it would be in the absence of the sports category contract. Thus, the sports-category-constraint contract affects the number of remaining sellable impressions for the business-category constraint, which does not overlap the sports-category constraint.

We formalize this problem as follows. Assume that we wish to find out the number of remaining impressions for constraint t . Let i ($= 1, \dots, n$) denote other constraints and j ($= 1, \dots, m$) denote attribute subspaces divided by all constraints. Define N_i and P_j as follows:

N_i : number of impressions contracted for constraint i ,
 P_j : estimated number of page views for attribute subspace j .

Note that N_i is the total sum of the impressions for all contracts which specify constraint i . In order to formulate the current problem as an instance of the “transportation problem”, we equalize the sum of N_i and the sum of P_j , by introducing dummy constraint $n + 1$ and dummy attribute subspace $m + 1$, and letting N_{n+1} be $\sum_{j=1}^m P_j$ and P_{m+1} be $\sum_{i=1}^n N_i$. We assume that constraint $n + 1$ contains every attribute subspace j and attribute subspace $m + 1$ is contained in every constraint i . We define S and Q as follows:

$$S \stackrel{\text{def}}{=} \{(i, j): \text{constraint } i \text{ contains attribute subspace } j\}, \quad (9)$$

$$Q \stackrel{\text{def}}{=} \{j: \text{constraint } t \text{ contains attribute subspace } j\}. \quad (10)$$

Let $x_{i,j}$ be the number of scheduled impressions for $(i, j) \in S$. Then, the problem is described as follows.

Problem 2. Minimize

$$\sum_{j \in Q} \sum_{i: (i,j) \in S, i \neq n+1} x_{i,j} + \sum_{i: (i,m+1) \in S, i \neq n+1} 2x_{i,m+1} \quad (11)$$

under the conditions

$$\sum_{j: (i,j) \in S} x_{i,j} = N_i \quad \text{for } i = 1, \dots, n + 1, \quad (12)$$

$$\sum_{i: (i,j) \in S} x_{i,j} = P_j \quad \text{for } j = 1, \dots, m + 1, \quad (13)$$

$$x_{i,j} \geq 0 \quad \text{for } (i, j) \in S. \quad (14)$$

left node to a right node is drawn for each $(i, j) \in S$. An arrow with a coarse broken line, a continuous line and a fine broken line indicates that the unit cost is 0, 1 and 2, respectively. Black nodes indicates that the corresponding attribute subspaces belong to Q . A solution for the problem is represented by an assignment of a number of scheduled impressions to each arrow. Arrows with bold lines are the ones to which a non-zero number of scheduled impressions is assigned in the optimal solution (the assigned number is parenthesized in the figure). For the arrows from the dummy-constraint node to the black nodes, the assigned numbers of scheduled impressions are 2,000 and 6,000. Therefore, the number of remaining sellable impressions for the business constraint is 8,000.

6. Empirical performance evaluation

6.1. Experimental set-up

We conducted large-scale simulation experiments to evaluate the effectiveness of some of the proposed methods. In our experiments, a probability model of the users was constructed, which specifies a conditional click rate for each ad and attribute value combination pair, and used this model to simulate the click-through behaviors. The model was constructed in the following fashion. First we decided on the number of ads and the number of attribute value combinations. In most of our experiments, we used a model with 32 ads and 128 attribute value combinations, which are divided into 32 clusters each containing 4 combinations. Within each cluster, the distribution was set to be in proportions of 1 : 2 : 3 : 4. The click rates were calculated as follows.

- (1) Determine the basic click rate pattern (p_j for $j = 0, \dots, 31$) according to the following table.

j	0	1–14	15–30	31
p_j	0.13	0.05	0.09	0.01

- (2) For each ad, randomly pick a multiplicative scaling factor a_j uniformly from $[0, 1]$.
- (3) For each cluster ad pair, Z_h, A_j , randomly pick an additive cluster noise, $\text{cnoise}(h, j)$, uniformly from $[-0.02, 0.02]$.
- (4) For each attribute value combination ad pair, i, j , pick an additive noise, $\text{noise}(i, j)$, uniformly from $[-0.005, 0.005]$.
- (5) Finally, for each cluster Z_h , for each attribute value combination $i \in Z_h$ and ad j , calculate the click rate $c_{i,j}$ as

$$c_{i,j} = a_j(p_{\text{index}(j,h)} + \text{cnoise}(h, j) + \text{noise}(i, j)).$$

Here $\text{index}(j, h)$ is set to be the remainder of dividing $j + h$ by m , so that the basic click rate pattern is translated (by one) for each cluster.

In each of the experiments reported here, we averaged over 5 campaign runs with each run consisting of 1,000,000 trials, using the above model, and with learning interval set at one of 3,125, 6,250, 12,500 or 25,000 trials. We set the desired number of impressions to be identical for all the ads.

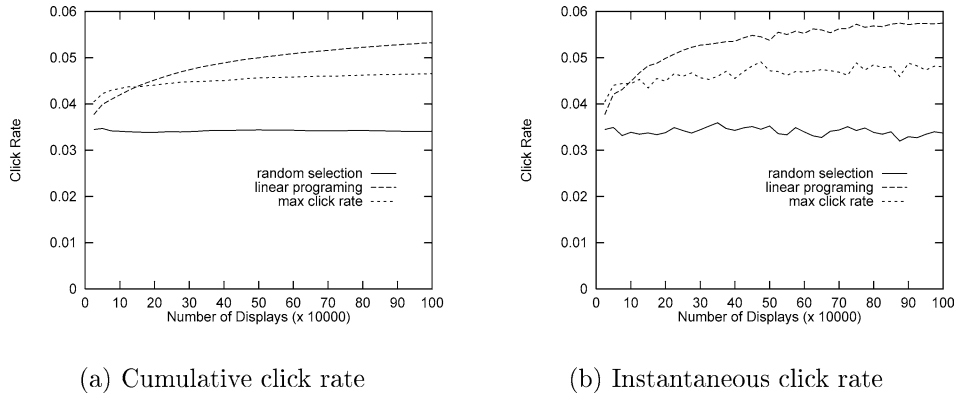


Figure 5. The total click rate of three methods.

The parameter values of our simulation experiment were set in a rather arbitrary fashion. In particular, the expected click-through rate of 3.5% in our model is rather high, as compared to the real world statistics—some report around 0.7% average click-through rate [DoubleClick, 6]. In general, the click through rates vary greatly depending on the site and the ads, so it is difficult to come up with a single simulation experiment that is “realistic”. Our primary intention in conducting these simulation experiments is to reveal general facts about the relative performance of competing strategies.

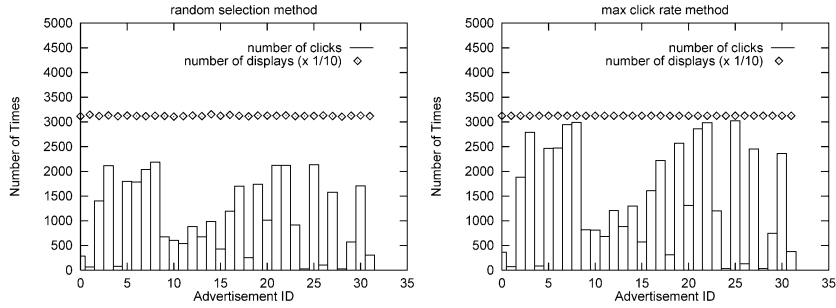
6.2. The effect of the linear programming approach

First we verified the effect of the linear programming approach by comparing its performance against that of two simple methods. One is the ‘random selection’ method, which always picks an ad uniformly at random, and the other is the ‘maximum selection’ method, which always picks the ad with the maximum estimated click rate for any attribute value combination among the ads that have not reached the desired number of impressions yet. The linear programming approach was enhanced with the ‘lower bounding’ technique described in an earlier section, and the learning interval was set at 3,125.

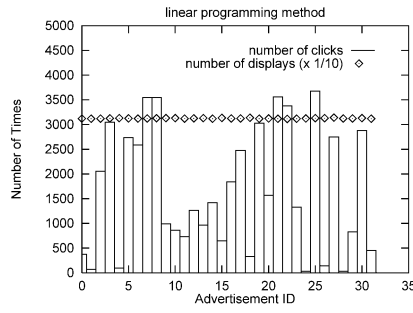
Figure 5 plots how the total click rate varies with each of the three methods. One can see that the linear programming method obtains the highest click rate of all three methods. Furthermore, for most ads, click rates of linear programming method are higher than those of other methods (Figure 6).

6.3. The effect of various trade-off measures

Figure 7 plots the total click rate for the linear programming method when it is combined with each of the additional measures we propose, clustering, Gittins Index, interior point method, and lower bounding, as well as the nothing-added (vanilla) version. Here the learning interval was set at 3,125. The graph on the left shows the cumulative click

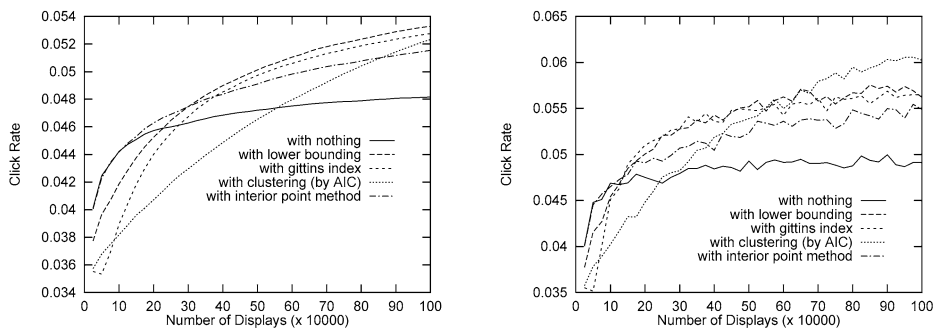


(a) The random sel. method (b) The maximum sel. method



(c) Linear programming method

Figure 6. (One tenth of) the number of impressions and number of clicks for each advertisement.



(a) Interval:3125, Cumulative click rate (b) Interval:3125, Instantaneous click rate

Figure 7. The effect of various measures on click rate.

Table 2. Final cumulative/instantaneous click rate (%) achieved by each of the five methods.

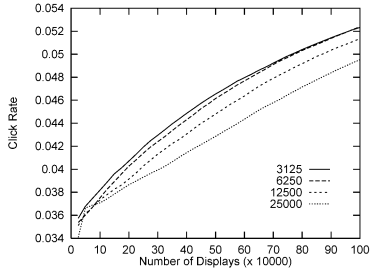
Final click rate	Cumulative	Instantaneous
Vanilla	4.82% (± 0.22)	4.91% (± 0.26)
Clustering	5.23% (± 0.08)	6.02% (± 0.22)
Gittins Index	5.28% (± 0.14)	5.65% (± 0.16)
Interior point	5.15% (± 0.14)	5.49% (± 0.36)
Lower Bounding	5.33% (± 0.11)	5.62% (± 0.17)

rate, while the one on the right shows the instantaneous click rate (averaged over the past 250,000 trials).

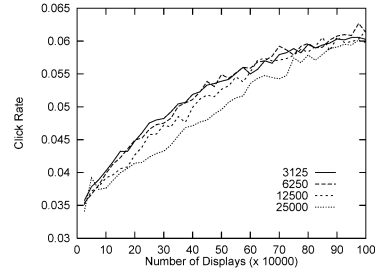
The same general tendency is found for each of the four additional measures—that in the beginning the click rate tends to be lower than the vanilla version (except the interior point method), but as the session progresses the instantaneous click rate quickly takes over and eventually the total cumulative click rate is improved by a significant margin. Note here that a session in these experiments consisted of a million trials, which correspond to the amount of transactions in a week or less at any major Internet provider. Table 2 summarizes the final (cumulative and instantaneous) click rate achieved by each of these methods, where we indicate the 95% confidence interval for each figure. It is clear that each of the four measures beats the vanilla version. Note that, with all but the interior point method, the difference in performance is statistically significant. As for the relative ordering among the four measures with respect to the overall performance, the results of these experiments stop short of being statistically significant.

We see in these experimental results that the average click-through rate of 3.5% achieved by the random selection method is improved to roughly 4.8% by the linear programming approach (without any additional techniques), and to approximately 5.3% by the versions with lower bounding and Gittins Index. These figures represent 37% and 51% increase in click-through rates, respectively. It is interesting to note that these figures are consistent with what other authors have observed. For example, Chickering and Heckerman [4] who applied a linear programming based ad targeting method in a real world setting, report a 30% increase in the total click-through rate.

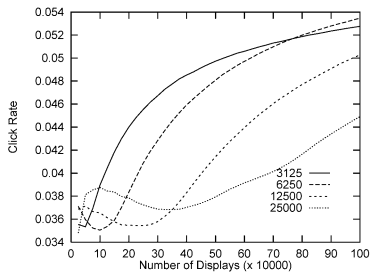
While some aspects of the simulation experiment are unrealistic (e.g., the average click through rate of 3.5% is considerably higher than the the average in the industry of 0.7% recently reported by DoubleClick [6]), the above observation is suggestive of the possibility that the relative improvement in click-through rate achieved by the various targeting methods may be relatively insensitive to these particulars, and will hold in a real world deployment. We also note that part of the revenue of an advertisement agency depends on the click rate. For example, ebiz101 (http://www.ebiz101.com/banner_ad_rates.htm) reports that Advertising.com charges 0.5–1.0 dollar per one click. Assuming that the relative performance of the various methods remains the same in real world deployment, the click-through rate can be increased by 37% and 51%, respectively by the vanilla LP method and the LP methods with lower bounding and Gittins Index, which would translate to significant increases in revenue.



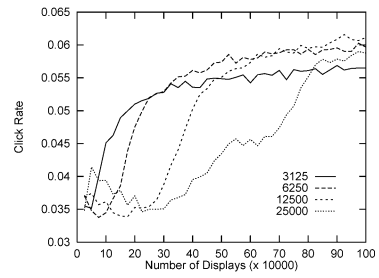
(a) Clustering, Cumulative click rate



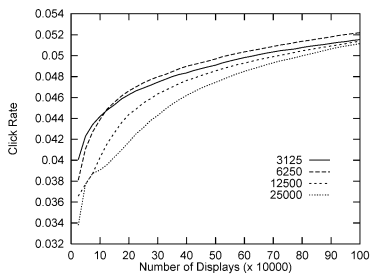
(b) Clustering, Instantaneous click rate



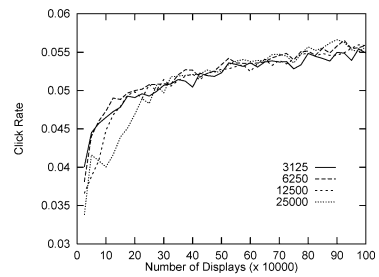
(c) Gittins index, Cumulative click rate



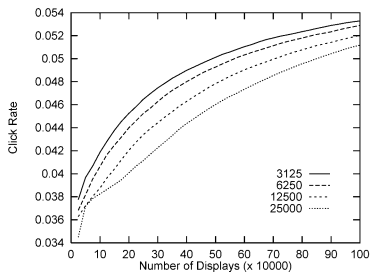
(d) Gittins index, Instantaneous click rate



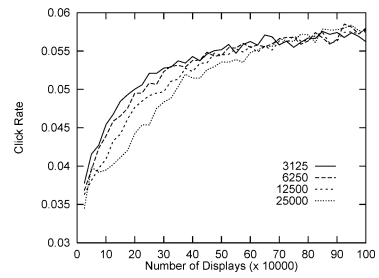
(e) Interior point method, Cumulative click rate



(f) Interior point method, Instantaneous click rate

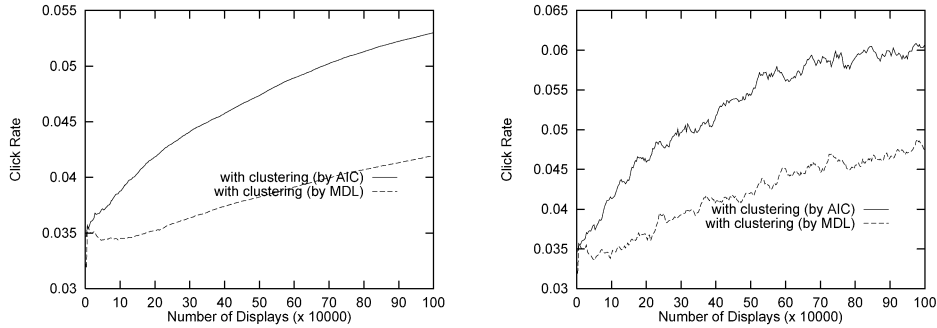


(g) Lower bounding, Cumulative click rate



(h) Lower bounding, Instantaneous click rate

Figure 8. Effect of learning interval on click rate.



(a) Learning interval:3125, Cumulative (b) Learning interval:3125, Instantaneous

Figure 9. Comparison between MDL and AIC.

Next, we examine the relative sensitivity of the four measures to the learning interval. Figure 8 plots how the total click rate changes for four different learning intervals, 3125, 6250, 12500 and 25000. It is immediately noticeable that Gittins Index is extremely sensitive to the learning interval—this method appears to have two phases, exploration-like phase and exploitation-like phase, and longer intervals cause longer exploration-like phase. The length of exploration-like phase is considered to depend on the number of learning needed to search the optimal combination of variables having non-zero display probabilities. Note that only a small number of display probabilities are non-zero in LP solution unless lower bounding or an interior point are used. We can see that longer intervals make the final instantaneous click rate higher for the LP method using Gittins Index. In contrast, the semi on-line interior point method and the lower bounding method seem to be relatively insensitive to the learning interval. The characteristic of the clustering method is that although it is slowest at improving the click rate it eventually reaches the highest instantaneous click rate overall, as indicated by the slope of these curves. All in all, we conclude that the relative merits of the various additional measures are dependent on the exact conditions of deployment, such as the learning interval and the campaign length. The interior point method and lower bounding seem to be suitable for any learning interval and any campaign length, but clustering performs better when campaign length is long. Gittins Index is suitable for short learning interval or long campaign length, and performs very well when the both lengths are long.

Incidentally, it was found that while our clustering method based on AIC was effective, the analogous method based on MDL actually degraded the performance in terms of click rate in our experiments (see Figure 9).

As MDL is known to perform well for the pure *estimation* problem (often better than AIC), it is interesting that in the current setting involving both exploration and exploitation, AIC works much better.

Table 3. Computation time (in seconds) of various methods on SUN ULTRA 2 (CPU: 300 MHz, Memory: 256 MB).

#Ad \times #Kw	32×128	64×256	128×512	256×1024
Transport	1.45	18.69	215.16	3224.19
Affine	16.83	136.11	1542.56	16342.77
Speed-up	11.61	7.28	7.17	5.07
CLUSTER+	3.25	35.15	408.18	5240.79
CLUSTER	21.80	332.16	5833.28	–
Speed-up	6.71	9.45	14.29	–

6.4. Computational efficiency issues

Here we report on the computation time measurements we made on the proposed methods, using a SUN ULTRA 2 workstation (CPU: 300 MHz, Memory: 256 MB). Table 3 summarizes the results. Note that Gittins Index and the lower bounding method were omitted from consideration since they had almost no effect on computation time.

The learning method, using the solution for the transportation problem, was found to be quite efficient with running time (with the clustering off) 3 minutes and 35 seconds for a problem of size 128×512 , and 53 minutes and 44 seconds for 256×1024 . Provided that the number of attribute value combinations is somehow reduced by putting low frequency ones together in a pre-processing phase, this method should be practical.

We also compared the computational efficiency of the algorithm we employ for the transportation problem with the affine scaling method. As summarized in Table 3, the formulation as a transportation problem helps cut the computation time by as much as 10-fold, though the speed-up ratio becomes smaller for larger problems. Finally, we compared the computational requirement of our clustering algorithm CLUSTER+, with a more naive version (CLUSTER) of the algorithm, which compares *all* cluster pairs to determine the pair with the most decrease in the I -value in each iteration. It was found that the efficiency measure devised in CLUSTER+ is quite effective, and is more so for larger sized problems. (Note that the table shows figures for experiments using the solution for the transportation problem.)

6.5. Simulated display probabilities bounds for multi-impressions

We simulated the ad selection process of function ADSelect for multi-impressions, and attempted to find the appropriate probability upper bounds p_k in inequality (8) for various number k of ads to be displayed concurrently, when the queue's capacity length is 100 and the number of multi-impressions is 100 millions. For computing the bounds for a probability upper bound p , we assume that we have $\lceil 1/p \rceil$ ads and $\lfloor 1/p \rfloor$ of those ads has display probability p . We judged that the capacity length 100 is sufficient for p if the queue length does not exceed 90 in all 10 runs, where each run consists of 100 million multi-impressions and each multi-impression consists of random selections of ads according to the given display probability distribution over the ads. For each $k = 2, 3, \dots, 10$, we

Table 4. Simulated display probability upper bound for the queue of capacity length 100.

k	2	3	4	5	6	7	8	9	10
p_k	0.458	0.294	0.215	0.164	0.138	0.117	0.102	0.083	0.079

searched for the maximal p for which the capacity length 100 is sufficient by this criterion, using binary search between 0 and $1/k$.

The result is shown in Table 4. By setting these values to p_k in inequality (8), the queue's overflow probability can be made negligibly low, while the range of feasible solutions is not significantly affected.

7. Conclusions

Our simulation experiments have verified the effectiveness of some of the proposed methods, both in terms of the improvement achieved in the click through performance and computational feasibility. The proposed methods have been incorporated into an actual Internet advertisement delivery server [Langheinrich et al., 9], and await further evaluation in a real world deployment.

Acknowledgments

We thank the following people involved in the Internet ad server project for numerous comments and support: Mr. Shimojima, Mr. Sugawa, Mr. Hiroya, Dr. Koseki, Dr. Kamba, Mr. Langheinrich, Dr. Takada, Dr. Doi and Dr. Goto, all of NEC Corporation.

Notes

1. Note that, while this assumption is not always valid (for example, attribute value 'not afternoon' does not appear in the afternoon), it is a reasonable assumption when considering the average behavior.
2. Note that the optimal solution depends on the promised number of impressions for each ad. If ad 2 has been promised to be displayed twice as many times as ad 1, then the optimal solution is the one that always displays ad 2 for combination 1 and ad 1 for combination 2.
3. A related clustering method, for the problem of estimating a model of word co-occurrence, was proposed by Li and Abe [Li and Abe, 10]. Here we propose an improved efficient algorithm for a related problem of estimating a conditional probability model.
4. Accurately estimating the number of page views itself is an important issue, but this is not the focus of the present paper.

References

- [1] Abe, N. and A. Nakamura. (1999). "Learning to Optimally Schedule Internet Banner Advertisements." In *Proc. of the 16th International Conference on Machine Learning*, pp. 12–21.
- [2] Bradley, G., G. Brown, and G. Graves. (1977). "Design and Implementation of a Large Scale Primal Transshipment Algorithm." *Management Science* 24, 1–34.

- [3] Berry, D.A. and B. Fristedt. (1985). *Bandit Problems*. Chapman & Hall.
- [4] Chickering, D. and D. Heckerman. (2000). "Targeted Advertising with Inventory Management." In *Proc. of ACM Special Interest Group on E-Commerce (EC00)*, pp. 145–149.
- [5] Dantzig, G. (1963). *Linear Programming and Extensions*. Princeton University Press.
- [6] DoubleClick Inc. (2003). DoubleClick 2002 Full-Year Ad Serving Trends.
- [7] Feller, W. (1968). *An Introduction to Probability Theory and its Applications*, Vol. 1. Wiley, 3rd edition.
- [8] Gittins, J.C. (1988). *Multi-Armed Bandit Allocation Indices*. Chichester: Wiley.
- [9] Langheinrich, M., A. Nakamura, N. Abe, T. Kamba, and Y. Koseki. (1999). "Unintrusive Customization Techniques for Web Advertising." *Computer Networks* 31, 1259–1272.
- [10] Li, H. and N. Abe. (1998). "Word Clustering and Disambiguation Based on Co-Occurrence Data." In *Proceedings of COLING-ACL*, pp. 749–755.
- [11] Nakamura, A. (2002). "Improvements in Practical Aspects of Optimally Scheduling Web Advertising." In *Proc. of the 11th International World Wide Web Conference*, pp. 536–541.
- [12] Nakamura, A. and N. Abe. (1998). "Collaborative Filtering Using Weighted Majority Prediction Algorithms." In *Proc. of the 15th International Conference on Machine Learning*, pp. 395–403.
- [13] Pazzani, M. (1999). "A Framework for Collaborative, Content-Based and Demographic Filtering." *Artificial Intelligence Review* 13(5–6), 393–408.
- [14] Quinlan, R. (1993). *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann.
- [15] Rissanen, J. (1978). "Modeling by Shortest Data Description." *Automatica* 14, 37–38.
- [16] Sakamoto, Y., Y. Ishiguro, and M. Kitagawa. (1986). *Akaike Information Criterion Statistics*. Dordrecht: Reidel.
- [17] Tomlin, J. (2000). "An Entropy Approach to Unintrusive Targeted Advertising on the Web." *Computer Networks* 33, 767–774.
- [18] Ye, Y. (1997). *Interior Point Algorithms: Theory and Analysis*, Wiley-Interscience Series in Discrete Mathematics and Optimization. New York: Wiley.