Delivering Guaranteed Display Ads under Reach and Frequency Requirements

Ali Hojjat* and **John Turner**
Paul Merage School of Business
University of California Irvine
hojjats@uci.edu, john.turner@uci.edu

Suleyman Cetintas and **Jian Yang**
Advertising Sciences Group
Yahoo Labs, Sunnyvale, CA
cetintas@yahoo-inc.com, jianyang@yahoo-inc.com

Abstract

We propose a novel idea in the allocation and serving of online advertising. We show that by using predetermined fixed-length streams of ads (which we call *patterns*) to serve advertising, we can incorporate a variety of interesting features into the ad allocation optimization problem. In particular, our formulation optimizes for representativeness as well as user-level diversity and pacing of ads, under reach and frequency requirements. We show how the problem can be solved efficiently using a column generation scheme in which only a small set of best patterns are kept in the optimization problem. Our numerical tests suggest that with parallelization of the pattern generation process, the algorithm has a promising run time and memory usage.

Introduction

Efficient serving of advertising is a key problem and a dominant source of revenue for online publishers. A large publisher may have hundreds of millions of page visits every day, and tens of thousands of concurrent advertising campaigns to manage. A *guaranteed* contract typically demands a certain number of ad impressions to be shown in certain slots on specific pages of the publisher's website. A *targeted* campaign further requires the ad to be shown only to users of certain demographic groups (e.g. age, gender, income level, location) and/or behavioral attributes (e.g. shopping). User arrivals, in aggregate, follow certain patterns which enables the publisher to forecast the supply of impressions and sell guaranteed advertising campaigns well in advance. Over short time intervals, however, the arrival of each user type is a lot less predictable. Even a few percent improvement in drawing the "correct" ad for each slot on the web page that each user sees can improve publisher revenues by tens of millions of dollars, increase advertising efficiency and return on investment for advertisers, and enhance user experience.

In this paper, we propose and examine a new idea for serving targeted display ads in the guaranteed display marketplace, namely *serving with patterns*. Upon the first visit over the serving period, we assign each user a predetermined fixed-length stream of ads, called a *pattern*, which is drawn from an existing pool of patterns according to the solution of our optimization problem. Then, in later visits, the user will be shown the exact sequence of ads in the assigned pattern. Further visits beyond the pattern length are treated as surplus ad inventory and served in a secondary sales channel (e.g. ad auctions in the non-guaranteed marketplace) until the next serving period. We show that the resulting formulation, although seemingly intractable, can be solved rather efficiently using a column generation approach. Furthermore, our formulation is capable of explicitly modeling all or any of the following concerns related to serving guaranteed display ads:

- **Representativeness:** The publisher should not try to deliver an entire contract to a small (potentially easy to serve) subgroup of targeted users. To capture a notion of *fair* allocation in the model, it is often assumed that the advertiser would like to be matched with a somewhat *equal mix* of targeted demographics. See (Ghosh et al. 2009) for a more elaborate discussion.

- **Reach and Frequency:** Advertisers can specify the number of unique individuals who should see their ad over the campaign period (*reach*), and also the minimum/maximum number of times each user should see their ad over the campaign period (*frequency*). This generalizes the commonly-practiced, and more widely-studied, idea of *impression*-based contracts that simply require a certain number of ad impressions to be shown to a targeted audience. To the best of our knowledge, our model is the first to consider optimal scheduling of online advertising under explicit reach and frequency specifications. This added flexibility, we expect, will greatly enhance the efficiency and ROI of the advertiser's campaign.

- **User-level Pacing:** The advertiser or the publisher may like to maintain a desired spacing of each ad over time (e.g. uniform delivery over time) which will determine how quickly each user is re-exposed to the ad over the campaign duration. To the best of our knowledge, existing research that explicitly considers uniform delivery of campaigns at the time of optimizing ad allocation focuses on the cumulative impressions received by each campaign in aggregate and not at the user-level (Araman and Fridgeirsdottir 2010). In practice, uniform delivery is considered as an implied requirement and treated with

*Work completed while interning at Yahoo Labs.

run-time *frequency capping*[1] heuristics. Our model is capable of explicitly measuring the spacing of ads over time at the optimization stage and generating patterns with the desired pacing at the user level.

- **User-level Diversification:** The publisher would like each user to see a variety of (relevant) ads from different advertisers. So, unless requested by an advertiser, the publisher prefers to avoid showing the same ad too many times to the same user. Again, the diversity of ads within a pattern can be explicitly modeled when patterns are being generated. Moreover, we have the ability to restrict the number of competing campaigns in a pattern (e.g. require that a pattern cannot contain both Coke and Pepsi ads). This is a common practice in arranging TV commercials within each break (Bollapragada, Bussieck, and Mallik 2004).

We begin with a brief overview of the underlying mathematical problem and relevant notation. We then describe how patterns can be used to serve ads, provide details of our column generation algorithm, and show our numerical results.

Problem Statement

The structure of a typical ad allocation problem can be represented with a bipartite graph, as shown in Figure 1. Each partition of user impressions (e.g. based on website, position of ad on the webpage, user demographics and behavioral attributes) is modeled as a *supply node*, indexed by $i \in \{1, \ldots, M\}$ on the left, and each ad campaign (advertising contract) is modeled as a *demand node*, indexed by $j \in \{1, \ldots, N\}$ on the right. The *arcs* show the targeting criteria of the campaigns, i.e., which impressions are eligible to be served with ads from which campaigns. We use $\Gamma(j)$ to denote the set of all impressions i eligible for contract j, and $\Gamma(i)$ to denote the set of all eligible contracts j that can be delivered to an impression of type i. Each supply node i represents s_i impressions, or $\hat{s}_i$ unique users, over the planning horizon (Note that $\hat{s}_i \leq s_i$ since each user may arrive multiple times). Similarly, each campaign may demand a total of d_j impressions across all eligible supply, or a reach of r_j unique users, with each user being required to see the ad at least $q_j^{\min}$ and at most $q_j^{\max}$ times to be counted as reached. The problem is then to find the optimal fraction of impressions i that should be allocated to each contract j, denoted x_{ij}, so as to minimize under-delivery, maximize representativeness, and achieve the desired user-level pacing and diversity of ads.

Modeling the ad allocation problem as a bipartite graph is not new. Langheinrich et al. (1999) is among the first and they use a linear objective to maximize the total click-through rate. More recently, Turner (2012) uses a quadratic objective to spread impressions across viewer types and thus minimize the variance of the number of impressions served.

[1]Frequency capping simply restricts number of views of each ad per visitor within a specific period of time (e.g. 1hr, or 24hrs) so the user is not overexposed to an ad. Browser cookies are typically used to keep track of impression counts. Many online publishers allow campaign managers to control the frequency cap.

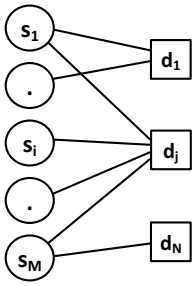

Figure 1: Example Bipartite Graph

Bharadwaj et al. (2012) consider the following impression-based formulation that minimizes a weighted average of non-representativeness (as an L2-norm penalty) as well as under-delivery:

$$\text{Minimize: } \frac{1}{2} \sum_{j, i \in \Gamma(j)} \frac{s_i V_j}{\theta_j} (x_{ij} - \theta_j)^2 + \sum_j p_j u_j \tag{1a}$$

$$\text{s.t. } \sum_{i \in \Gamma(j)} s_i x_{ij} + u_j \geq d_j \quad \forall j \tag{1b}$$

$$\sum_{j \in \Gamma(i)} x_{ij} \leq 1 \quad \forall i \tag{1c}$$

$$x_{ij},\ u_j \geq 0 \quad \forall i, j \tag{1d}$$

Demand constraint (1b) requires the total number of impressions allocated to each contract j to exceed its demand d_j, or otherwise we have an under-delivery of u_j impressions. Supply constraint (1c) implies that we cannot allocate more than 100% of supply from each node i. Each contract has an under-delivery penalty p_j per impression, and a relative importance weight V_j. Parameter $\theta_j = d_j \big/ \big(\sum_{i \in \Gamma(j)} s_i \big)$ denotes the ratio between the contract's demand and its total eligible supply. By definition, in a perfectly representative allocation, each contract should grab exactly a θ_j–proportion of each eligible supply pool, and therefore the deviation from θ_j is (quadratically) penalized in (1a). Using duality theory, (Bharadwaj et al. 2012) develop an efficient iterative algorithm, referred to as SHALE, for solving a problem with the above structure. Similar formulations of the problem and further discussion can be found in (Nakamura and Abe 2005; Yang et al. 2010).

In the following section we demonstrate how the above formulation can be modified to incorporate a variety of interesting features, when we serve ads using *patterns*. Note our main idea (of using patterns) can be incorporated into any math program with similar steps described below.

Serving ads using Patterns

We define a *serving pattern* as a finite permutation of ads. A particular campaign may show up multiple times at different points in the serving pattern, and the pattern should not necessarily contain all campaigns. A few examples of patterns composed of three campaigns {A,B,C} are shown in Figure 2. The first pattern has 6 slots, and an equal number of each ad are spread uniformly throughout the pattern. The next patterns have a length of 8, with campaign C appearing

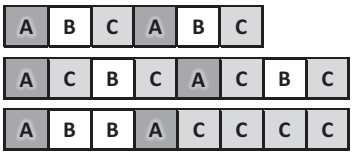

Figure 2: Example of patterns with 3 campaigns {A,B,C}

twice as often as campaigns A or B. The second pattern illustrates uniform pacing (assuming arrivals are also uniform over time), whereas the third pattern delivers campaigns B and C repeatedly to strengthen user recall.

Upon the first arrival in the serving period, each individual user is assigned a particular serving pattern, and upon his/her k^{th} arrival will be shown the ad in the k^{th} slot of the assigned pattern. The goal is then to find a handful of well-paced patterns and a representative user assignment such that reach and frequency requirements are met.

Let $\mathcal{N}$ denote the entire set of possible patterns of different lengths and structure. For each pattern $n \in \mathcal{N}$, we know the exact number of times each contract j appears in the pattern (i.e. its frequency), denoted a_{jn}, and we can easily tell if a_{jn} is within the desired frequency range $\left(q_j^{\min}, q_j^{\max}\right)$, which we denote using the binary indicator b_{jn}. The quality of the pacing and diversity of ads within a pattern is captured by the cost parameter π_n whose precise definition will become clear when we describe our pattern generation problem. Let v_{in} denote the number of times pattern n is assigned to users from supply node i. Therefore, $\sum_n a_{jn}v_{in}$ gives the total impressions of j shown to supply pool i, and $\sum_n b_{jn}v_{in}$ is the number of unique users in i who have seen j within the desired frequency range. Realizing that $x_{ij} = \frac{1}{s_i}\sum_n a_{jn}v_{in}$, we can formulate our ad allocation problem in similar form to (1) as:

$$\text{Minimize:} \quad \frac{1}{2}\sum_{j,i\in\Gamma(j)} \frac{s_i V_j}{\theta_j}\left(\frac{\sum_n a_{jn}v_{in}}{s_i} - \theta_j\right)^2 + \sum_j p_j u_j + \sum_{i,n}\pi_n v_{in} \tag{2a}$$

$$\text{s.t.} \quad \sum_{n,i\in\Gamma(j)} b_{jn}v_{in} + u_j \geq r_j \quad \forall j \tag{2b}$$

$$\sum_n v_{in} \leq \hat{s}_i \quad \forall i \tag{2c}$$

$$v_{in},\ u_j \geq 0 \quad \forall i,j,n \tag{2d}$$

Notice that we use the cost parameter π_n to explicitly penalize non-smooth and/or non-diverse delivery at the user level in the objective function. The demand constraint (1b), or equivalently $\sum_{n,i\in\Gamma(j)} a_{jn}v_{in} + u_j \geq d_j$, is replaced with the reach and frequency constraint (2b) which requires the contract to be served with at least r_j unique users seeing the ad within the correct frequency range. The shortfall u_j is penalized in the objective. Note that under-delivery u_j is now measured in terms of reach (and not impressions). The supply constraint (2c) ensures that the total number of patterns assigned to pool i cannot exceed the number of unique users in i (since each user is assigned a single pattern). Here, representativeness is still measured in terms of impressions, but one could easily express it in terms of reach as $\sum_{j,i\in\Gamma(j)} \frac{\hat{s}_i V_j}{\hat{\theta}_j}\left(\frac{\sum_n b_{jn}v_{in}}{\hat{s}_i} - \hat{\theta}_j\right)^2$, where $\hat{\theta}_j = r_j \big/ \left(\sum_{i\in\Gamma(j)} \hat{s}_i\right)$ is the ratio between the desired reach of contract j and the available number of unique users across all targeted users.

A key assumption in the above model is that each user, over the serving period, will generate at least as many impressions as there are in the serving pattern assigned to him/her. Only then is it guaranteed that the user will see each ad j the a_{jn} number of times (where s/he is counted as reached if $b_{jn} = 1$) as we planned when we solved (2). To this end, we assume that users can be further classified according to their browsing behavior. All users of the same visit type, $w \in \mathcal{W}$, share a common probability distribution, $\phi_w(k)$, that gives the probability of such a user generating exactly k impressions over the serving period. We can then say that each user of type w will make at least $L_w(\rho) = \Phi_w^{-1}(1-\rho)$ visits with probability ρ. With a reasonably high ρ, we can use the resulting $L_w(\rho)$ (from now on referred to in short as L_w) as the appropriate pattern length for a user of type w. To reflect this in our model, each supply node needs to be partitioned further based on the user visit types w, and we also need to maintain a separate pool of patterns, $\mathcal{N}_w$, in order to ensure that each user of type w is assigned a pattern of appropriate length L_w. The updated formulation is provided in the following section when we formlize our master problem. We should also point out that further impressions that the user generates beyond L_w are assumed to be served in a secondary market (e.g. ad auctions in the non-guaranteed marketplace) and are therefore ignored in our formulation.

Column Generation

Indeed, generating all possible patterns, storing them in memory, and solving an optimization problem that requires as input the entire possible pattern set $\mathcal{N}$ is unmanageable. However, this issue can be handled rather efficiently using a column generation scheme[2]. The idea is to start with a small pool of patterns, solve an assignment problem of type (2), and then use the optimal primal/dual solution to generate new patterns that can improve the current solution. We will then add these improving patterns to the pool and solve the assignment problem again, and repeat the procedure until no improving pattern can be found (i.e., full convergence to the optimal solution), or the improvement in the objective function seems negligible. In the following, we describe the two core steps of the column generation algorithm.

[2]For the theory of column generation, the reader may refer to (Desaulniers, Desrosiers, and Solomon 2005). Other applications of column generation in the allocation of online advertising can be found in (Abrams et al. 2008; Walsh et al. 2010).

Pattern Assignment

For any given set of pattern pools $\{\mathcal{N}_w : w \in \mathcal{W}\}$, the optimal assignment of patterns to users is given by the following quadratic program, which we refer to as the *Master Problem*:

$$\Psi := \text{Min} \quad \frac{1}{2} \sum_{\substack{j,w,\\ i\in\Gamma(j)}} \frac{s_{wi}V_j}{\theta_j} \left(\frac{1}{s_{wi}} \sum_{n\in\mathcal{N}_w} a_{jn} v_{win} - \theta_j \right)^2 + \sum_j p_j u_j + \sum_{\substack{w,i,\\ n\in\mathcal{N}_w}} \pi_n v_{win} \tag{3a}$$

$$\text{s.t.} \quad \sum_{\substack{w,i\in\Gamma(j),\\ n\in\mathcal{N}_w}} b_{jn} v_{win} + u_j \geq r_j \quad \forall j \tag{3b}$$

$$\sum_{n\in\mathcal{N}_w} v_{win} \leq \hat{s}_{wi} \quad \forall w, i \tag{3c}$$

$$v_{win},\ u_j \geq 0 \quad \forall j, w, i, n \in \mathcal{N}_w \tag{3d}$$

Here, s_{wi} and $\hat{s}_{wi}$ are, respectively, the total number of impressions and unique users of visit type w in supply node i, and v_{win} is the number of such users that should be served with pattern n (of appropriate length L_w selected from $\mathcal{N}_w$). Once the above is solved, we can use $v^*_{win}/\hat{s}_{wi}$ as the probability that pattern $n \in \mathcal{N}_w$ should be assigned to a user of type (w, i) upon his/her first visit.

Note that by maintaining separate pattern pools based on length, $\{\mathcal{N}_w : w \in \mathcal{W}\}$, we ensure that a pattern of the correct length L_w is assigned to a user of visit type w. However, the set of contracts appearing in each pattern may not be entirely eligible for all impression types i, since in general, $\Gamma(i) \neq \Gamma(i')$ if $i \neq i'$. At the time of serving we should draw patterns of not only the correct length, but also containing only (some subset of) the eligible contracts $j \in \Gamma(i)$ for the arriving impression (w, i). This may suggest that we need to maintain a separate pattern pool for each (w, i); but in fact, this is unnecessary. The terms $\sum a_{jn} v_{win}$ and $\sum b_{jn} v_{win}$ in the objective function (3a) and demand constraint (3b) are only evaluated for neighboring (i, j) pairs. Therefore, ineligible contracts $j \notin \Gamma(i)$ in a pattern are counted as lost impressions if the pattern is assigned to a user of type (w, i). Therefore, with a scarce supply of unique users and guaranteed impressions, we expect the math program to naturally avoid such assignments and always find (or generate, in the following iteration) a "fully eligible" pattern to replace any "partially eligible" pattern. In the end, we expect $v^*_{win} = 0$ if the set of contracts in pattern n are not fully eligible for i. This is important, since if we were forced to use separate pattern pools for each user type (w, i), we would significantly (and unnecessarily) increase memory usage and the scale of the problem. To see why, note that some patterns may be relatively short (e.g., 5 or 10 slots in length) and therefore contain only a few contracts. Moreover, the sets $\Gamma(i)$ and $\Gamma(i')$ for $i \neq i'$ can have significant overlap. Therefore, we would end up with many duplicate patterns composed of contracts $j \in \Gamma(i) \cap \Gamma(i')$ if we used separate pattern pools $\mathcal{N}_{wi}$ and $\mathcal{N}_{wi'}$.

Let α_j and β_{wi} denote the (non-negative) dual multipliers of the reach and supply constraints (3b) and (3c), respectively. The reduced cost of each variable v_{win} can be derived by constructing the full Lagrangean of the problem and taking its derivative with respect to v_{win}. We have:

$$\begin{aligned} \mathcal{L} \ = \ & \frac{1}{2} \sum_{\substack{j,w,\\ i\in\Gamma(j)}} \frac{s_{wi}V_j}{\theta_j} \Big(\frac{1}{s_{wi}} \sum_{n\in\mathcal{N}_w} a_{jn} v_{win} - \theta_j \Big)^2 \\ & + \sum_j p_j u_j \ + \sum_{\substack{w,i,\\ n\in\mathcal{N}_w}} \pi_n v_{win} \\ & + \sum_j \alpha_j \Big(r_j - u_j - \sum_{\substack{w,i\in\Gamma(j),\\ n\in\mathcal{N}_w}} b_{jn} v_{win} \Big) \\ & + \sum_{w,i} \beta_{wi} \Big(\sum_{n\in\mathcal{N}_w} v_{win} - \hat{s}_{wi} \Big), \end{aligned} \tag{4}$$

from which we have:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial v_{win}} = & \sum_{j\in\Gamma(i)} \frac{V_j}{\theta_j} \Big(\frac{1}{s_{wi}} \sum_{n\in\mathcal{N}_w} a_{jn} v_{win} - \theta_j \Big) a_{jn} \\ & + \pi_n - \sum_{j\in\Gamma(i)} \alpha_j b_{jn} + \beta_{wi}. \end{aligned} \tag{5}$$

The stationarity condition for an optimal solution requires (5) to be zero for all basic variables (i.e., patterns that are used in the solution), and positive for all non-basic variables (suggesting that patterns which are unused would increase and worsen the objective function if they were used). However, we might be able to find (or more precisely, construct) a pattern k, not currently existing in any of the pattern pools, with a_{jk}, b_{jk}, and π_k values that renders the reduced cost function (5) negative. Adding such a pattern to the pattern pool is then expected to improve the optimal solution when (3) is re-solved. We use this property to formulate our pattern-generation model.

Pattern Generation

Let v^*_{win}, α^*_j, and β^*_i denote the optimal primal and dual solutions to (3) given the current set of pattern pools $\mathcal{N}_w$, and let $x^*_{wij} = \frac{1}{s_{wi}} \sum_{n\in\mathcal{N}_w} a_{jn} v^*_{win}$ denote the current impression-allocation plan. For each user type (w, i), the following model, which we refer to as the *subproblem*, will generate a pattern of minimum reduced cost:

$$\psi_{wi} := \text{Min} \sum_{j\in\Gamma(i)} \frac{V_j}{\theta_j} \left(x^*_{wij} - \theta_j \right) \widehat{a}_j - \sum_{j\in\Gamma(i)} \alpha^*_j \widehat{b}_j + \widehat{\pi}(\cdot) \tag{6a}$$

$$\text{s.t.} \quad \sum_{j\in\Gamma(i)} \widehat{a}_j = L_w \tag{6b}$$

$$q_j^{\min} \widehat{b}_j \leq \widehat{a}_j \leq q_j^{\max} \quad \forall j \in \Gamma(i) \tag{6c}$$

$$\widehat{a}_j : \text{Integer},\ \widehat{b}_j \in \{0, 1\}, \quad \forall j \in \Gamma(i) \tag{6d}$$

Our decision variables, and the cost metric (which depends on our decision variables) are marked with a caret ($\widehat{\ }$). The integer variable $\widehat{a}_j$ counts the number of times campaign j appears in the pattern. The knapsack constraint (6b) ensures that the correct length L_w of the pattern is used and fully utilized. The constraint set (6c) ensures that $\widehat{a}_j$ is at most the maximum frequency required by the contract j (as we

would otherwise waste an ad slot in the pattern). Finally, the binary variable $\widehat{b}_j$ indicates whether or not the pattern being generated meets the minimum frequency requirement for campaign j (Note that $\widehat{b}_j$ has a negative coefficient in the objective function, and so $\widehat{b}_j = 1$ is always desirable).

If the optimal values of all (w, i) subproblems yield $\psi^*_{wi} + \beta^*_{wi} \geq 0$, i.e., the reduced costs (5) are all non-negative, then no improving pattern exists. In that case, the current solution to the master problem (3) is optimal and the current set of patterns, contained in the pattern pools $\{\mathcal{N}_w : w \in \mathcal{W}\}$, comprise the basic subset of $\mathcal{N}$. If, however, we find $\psi^*_{wi} + \beta^*_{wi} < 0$ for any (w, i), we will add the constructed pattern to the pool $\mathcal{N}_w$ with $a_{jn} = \widehat{a}_j$, $b_{jn} = \widehat{b}_j$, and $\pi_n = \widehat{\pi}(\cdot)$. For increased memory efficiency, one may also remove all patterns that are unused in the current solution (although the removed patterns may enter the pool again in later iterations of the master problem).

It is noteworthy to point out that we do not need to solve the above subproblems for all (w, i) before returning to the master problem. Even with a single improving pattern we are guaranteed to improve upon the optimal value in (3). Therefore, one may solve the subproblems, in some clever ordering, until a certain number of new patterns are found, and then return to re-solve the master problem. Moreover, the above subproblems are independent from one another, and therefore the pattern construction step is parallelizable.

To initialize the algorithm, we can use any heuristic to generate an initial set of patterns, or we may set $\alpha_j = 0$, $\beta_{wi} = 0$, $x_{ij} = 0$ (which is primal/dual feasible) and solve as many subproblems as needed to construct a desired number of patterns with negative reduced cost. In practice, ad schedules would need to be re-optimized periodically (and rather frequently) with updated supply forecasts and contract lists. As long as the problem parameters (i.e. the list of guaranteed contracts and supply forecasts) do not change drastically from one period to the next, we expect many of the patterns from the previous period to be good candidates for initializing the pattern pools.

Note that the overall structure of (6) is similar to a knapsack problem (with complicating frequency capping constraints (6c)). However, the degree of difficulty of (6) highly depends on the functional choice for the cost function $\pi(\cdot)$. Recall that $\pi(\cdot)$ is a cost function that measures the (lack of) quality of a pattern in terms of pacing and/or diversity of ads. Assuming that diversity and pacing metrics are separable, we can write $\pi(\cdot)$ as a weighted sum of diversity and pacing metrics: $\pi(\cdot) = \lambda_d \pi_d(\cdot) + \lambda_p \pi_p(\cdot)$. A diversity-seeking cost function may take the form:

$$\pi_d(\cdot) = -\sum_j \widehat{I}_j, \tag{7}$$

which simply encourages including more campaigns in the pattern. The binary variables $\widehat{I}_j$ indicate whether campaign j appears in the pattern at all. The Big-M constraints of the form $\widehat{a}_j \leq L_w \widehat{I}_j$ should also be added to the subproblem to enforce $\widehat{a}_j = 0$ whenever $\widehat{I}_j = 0$.

A uniform-pacing cost function may take the following form (adapted from (Kubiak and Sethi 1991)):

$$\pi_p(\cdot) = \sum_{j \in \Gamma(i)} \sum_{k=1}^{L_w} \left(\bar{z}_{jk} - \frac{k}{L_w} \widehat{a}_j \right)^2, \tag{8}$$

where $\bar{z}_{jk}$ is the number of times campaign j appears in the first k slots of the pattern. In order to incorporate (8) into (6) we also need an additional set of binary decision variables z_{jk} that indicate whether campaign j is placed in the k'th slot of the pattern. Furthermore, we need constraints $\sum_j z_{jk} = 1$, so exactly one ad is placed in each slot k of the pattern; $\bar{z}_{jk} = \sum_{r=1}^{k} z_{jr}$ to model the cumulative relationship; and finally $\widehat{a}_j = \bar{z}_{jL_w}$. The above pacing metric simply assumes that with uniform pacing, the cumulative count $\bar{z}_{jk}$ should grow linearly with slope $\widehat{a}_j / L_w$ throughout the pattern length.

Finally, if we have any set of competing campaigns, $\mathcal{C}$, which should not be shown to the same user (e.g., Coke and Pepsi), we can add a constraint of the form $\sum_{j \in \mathcal{C}} \widehat{I}_j \leq 1$ so at most one of the competing campaigns is put into the pattern.

Bollapragada, Bussieck, and Mallik (2004) examine more involved formulations for the uniform arrangement of TV advertising. They describe the problem as "placing balls of different colors into a fixed number of slots such that balls of the same color are as equally-spaced as possible". In their model, the number of balls of each color ($\widehat{a}_j$ in our model) is considered as given. Their performance test on a variety of formulations shows that a (sub-optimal) greedy approach, based on the formulation of (Kubiak and Sethi 1991), is the only viable way of solving the problem with more than 50 slots (L_w) in a reasonable amount of time. Note that our model has a higher degree of complexity since the subproblem should also figure out the optimal *number* of balls of each color ($\widehat{a}_j$) at the *same time* as finding the optimal uniform arrangement.

One practical approach is to postpone the arrangement of ads within the patterns until the assignment problem is fully solved. That is, we no longer evaluate the best possible arrangement of patterns in terms of pacing at the time of generating the patterns. Instead, we apply an exact or greedy algorithm, e.g. from (Bollapragada, Bussieck, and Mallik 2004), only on the surviving patterns in the optimal solution (with already-known frequencies a_{jn}) to evenly pace the ads before streaming the patterns to user visits.

Numerical Experiments

Prior work in exposure-based guaranteed advertising is impression-based; that is, it assumes publishers do not differentiate between serving 2 impressions to 1 person, or 1 impression each to 2 people. Consequently, there are no existing benchmarks for comparing the performance of our algorithm. We tested the algorithm on randomly-generated graphs that we constructed in such a fashion to resemble appropriately-scaled[3] versions of real-world instances. For

[3]by matching the distribution of parameters (supply and demand values, density of links between nodes, and θ_j values).

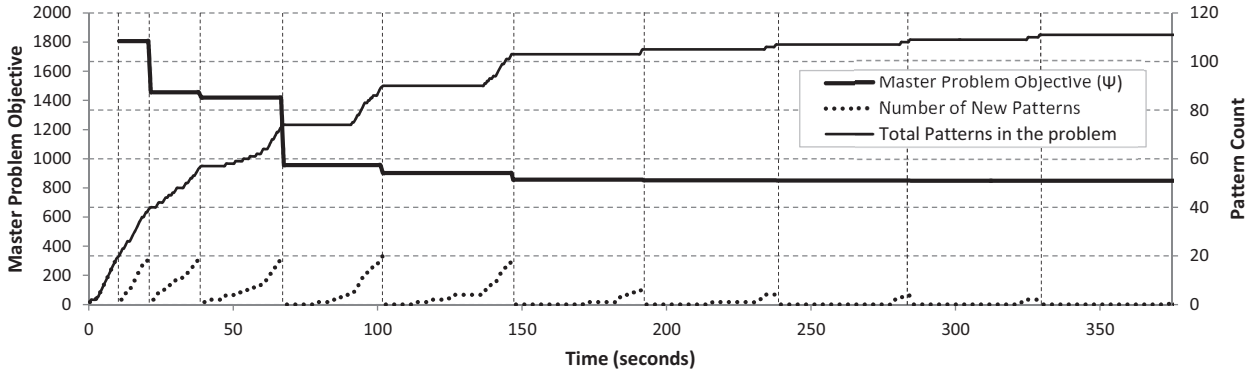

Figure 3: Performance of the Column Generation Algorithm on a Sample Graph

example, Figure 3 demonstrates the progress of the algorithm on a small graph with 40 demand nodes and 300 supply nodes. Each supply node was further partitioned into 3 subgroups with guaranteed visit lengths of $\{10, 20, 30\}$ impressions. There are approximately 4600 arcs in the graph. The horizontal axis shows time (in seconds). Each vertical dashed line shows an epoch where the master problem is solved, and the thick black curve tracks the optimal value of the master problem, denoted Ψ^*, that decreases each time the master problem is re-solved. In between the epochs, we solve the subproblems until (at most) 20 improving patterns were found. The dotted curves show the cumulative number of new patterns found during each epoch, and the solid thin curve shows the total number of patterns existing in the master problem. Throughout the process, we deleted old unused patterns to keep the total number of available patterns at each point in time from growing too quickly. We solved the subproblems in an ad-hoc (essentially random) order. We used a diversity-seeking metric of the form (7), and did not use a pacing metric (i.e. $\pi_p(\cdot) = 0$). We used the AMPL modeling language with CPLEX solver on a dual core i5 2.5GHz CPU with 8GB of RAM to carry out the experiments.

The master problem fully converged to the optimal solution after 10 iterations (6 minutes), at which point we solved all 900 subproblems to verify that no improving pattern existed. As we can see in Figure (3), the optimal value Ψ^* initially improves quickly, but the rate of improvement tapers off, becoming negligible beyond iteration 6 (2.5 minutes). Note that the subproblems are not being solved in parallel in our numerical experiment. With full parallelization, the full convergence could be attained in less than 1 minute. Moreover, there is a tradeoff between the number of iterations it takes for the master problem to converge and the maximum number of new patterns we aim for during each epoch. With no limit on the number of new patterns, the above example would converge in 4 iterations; however, 900 subproblems need to be solved in each iteration, and the total run time happens to be worse than 6 minutes.

Note that among the possible $O\left(10^{19}\right)$ patterns that can be constructed for this small instance, only 111 are used in the final solution[4]. This illustrates the power of column generation to isolate only the best patterns. This further shows that keeping a separate pattern pool for each (w, i) combination is highly inefficient. If we grouped patterns by (w, i), we would need to have at least $300 \times 3 = 900$ patterns in the solution (one per each $\mathcal{N}_{wi}$ to be able to access $\hat{s}_{wi}$).

Finally, we would like to point out that the improvement in the optimal value of the master problem is not guaranteed to be monotonically decreasing. For instance, the improvement in Ψ^* in iterations 3 and 5 was very low, whereas a number of patterns were found during iteration 4 which drastically improved the solution. Therefore, a termination criteria based on the absolute or relative improvement in Ψ^* should be used with great caution.

Concluding Remarks

In this paper, we proposed a novel idea for allocating and serving online advertising: using predetermined fixed-length streams of ads (which we call *patterns*). Our framework introduces a *user-level* perspective into the common *aggregate* modeling of the ad allocation problem. This enables us to incorporate a variety of features (that are typically modeled and analysed as separate problems in the literature) into a single optimization problem. In particular, our formulation can optimize for representativeness as well as user-level diversity and pacing of ads, under reach and frequency requirements. We showed that the problem can be solved efficiently using a column generation scheme in which only a (small) set of best patterns are kept in the problem. In each iteration, we solve a set of optimization problems to generate new patterns that improve the solution. Our preliminary numerical tests show that near-optimal solutions are attained rather quickly with a relatively small number of patterns. Furthermore, the run time can be drastically improved by parallelizing the pattern generation process.

[4]If we differentiate patterns based on the exact arrangement of ads within the pattern, we can construct $\sum_L 40^L = 1.15 \times 10^{48}$ patterns, given $L \in \{10, 20, 30\}$. If we differentiate only based on the number of times each campaign appears in the pattern, we can construct $\sum_L \sum_{c=1}^{L} \binom{40}{c}\binom{L-1}{c-1} \approx 3.16 \times 10^{19}$ patterns.

References

Abrams, Z.; Keerthi, S. S.; Mendelevitch, O.; and Tomlin, J. A. 2008. Ad delivery with budgeted advertisers: A comprehensive LP approach. *Journal of Electronic Commerce Research* 9(1).

Araman, V. F., and Fridgeirsdottir, K. 2010. A uniform allocation mechanism and cost-per-impression pricing for online advertising. *Working paper*.

Bharadwaj, V.; Chen, P.; Ma, W.; Nagarajan, C.; Tomlin, J.; Vassilvitskii, S.; Vee, E.; and Yang, J. 2012. SHALE: An efficient algorithm for allocation of guaranteed display advertising. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, 1195–1203. ACM.

Bollapragada, S.; Bussieck, M. R.; and Mallik, S. 2004. Scheduling commercial videotapes in broadcast television. *Operations Research* 52(5):679–689.

Desaulniers, G.; Desrosiers, J.; and Solomon, M. M. 2005. *Column generation*, volume 5. Springer.

Ghosh, A.; McAfee, P.; Papineni, K.; and Vassilvitskii, S. 2009. Bidding for representative allocations for display advertising. In *Internet and Network Economics*. Springer. 208–219.

Kubiak, W., and Sethi, S. 1991. A note on "level schedules for mixed-model assembly lines in just-in-time production systems". *Management Science* 37(1):121–122.

Langheinrich, M.; Nakamura, A.; Abe, N.; Kamba, T.; and Koseki, Y. 1999. Unintrusive customization techniques for web advertising. *Computer Networks* 31(11):1259–1272.

Nakamura, A., and Abe, N. 2005. Improvements to the linear programming based scheduling of web advertisements. *Electronic Commerce Research* 5(1):75–98.

Turner, J. 2012. The planning of guaranteed targeted display advertising. *Operations Research* 60(1):18–33.

Walsh, W. E.; Boutilier, C.; Sandholm, T.; Shields, R.; Nemhauser, G. L.; and Parkes, D. C. 2010. Automated channel abstraction for advertising auctions. In *AAAI*.

Yang, J.; Vee, E.; Vassilvitskii, S.; Tomlin, J.; Shanmugasundaram, J.; Anastasakos, T.; and Kennedy, O. 2010. Inventory allocation for oonline graphical display advertising. *arXiv preprint arXiv:1008.3551*.