# Optimal Delivery of Sponsored Search Advertisements Subject to Budget Constraints

Zoe Abrams
Yahoo!, Inc.
701 First Avenue
Sunnyvale, CA , USA
za@yahoo-inc.com

Ofer Mendelevitch
Yahoo!, Inc.
701 First Avenue
Sunnyvale, CA , USA
oferm@yahoo-inc.com

John A. Tomlin
Yahoo! Research
701 First Avenue
Sunnyvale, CA , USA
tomlin@yahoo-inc.com

## ABSTRACT

We discuss an auction framework in which sponsored search advertisements are delivered in response to queries. In practice, the presence of bidder budgets can have a significant impact on the ad delivery process. We propose an approach based on linear programming which takes bidder budgets into account, and uses them in conjunction with forecasting of query frequencies, and pricing and ranking schemes, to optimize ad delivery. Simulations show significant improvements in revenue and efficiency.

## Categories and Subject Descriptors

G.4 [**Mathematics of Computing**]: Mathematical Software—*Algorithm design and analysis*

## General Terms

Algorithms, Performance

## Keywords

column generation, sponsored search, budgets, advertising

## 1. INTRODUCTION

Search engine companies such as Yahoo!, Google, and MSN, earn millions of dollars each day by auctioning off advertisement slots. In addition to the bids, there are two essential sets of parameters of the system that contribute to this revenue—the distribution of query frequencies and the advertiser budgets.

The query frequencies limit the number of times the search engine can display its advertisers. Query frequencies are not under the control of the advertisers or the search engine. It is well known [17] that search engine query frequency distribution typically has a few queries with large volume (and large revenue and efficiency), and a very large number of queries with extremely low volume. Therefore, we overcome most of the uncertainty in query volumes by selecting a relatively small subset of queries whose near-term volumes are easy to forecast, yet still constitute a large amount of the overall revenue.

Advertisers or their agents, on the other hand, do have the ability to control their budgets. An advertiser's budget may constrain the number of times their ads appear, even when they have made a high bid on a query term. One might ask why they would wish to do so. There are several possible reasons, among them: protection against click-fraud, an over-all company advertising budget, and the desire to control the allocation of that budget between various media and campaigns. Whatever the reason, the search engine must determine which advertisers to display for which queries, given these constraints.

Pricing and ranking are additional parameters of the system that influence revenue. The VCG mechanism [19, 5, 10] can be applied, but in practice search engines predominantly use the *generalized second price* (GSP) auction (see Edelman et al. [9] for more details). Advertisers are ranked according to the product of the price they bid for receiving a click, and a quality score. Each is charged a price per click equal to the minimum they would have had to pay to maintain their rank.

The problem we consider is how to allocate advertisers to queries such that budget constraints are satisfied and efficiency (or revenue) is maximized. This problem is posed as a linear program that takes a global view, and coordinates advertiser spend across the chosen time-period, such as the next hour, or an entire day. With the combined knowledge of forecast query volumes, advertiser budgets, advertiser bids and the pricing and ranking algorithm, we formulate a comprehensive mathematical framework.

## 1.1 Related Work

Incorporating advertiser budgets into the marketplace design is recognized as crucial, and a growing amount of research addresses this subject. Several recent papers have considered the effect of budgeted advertisers specifically within the context of internet keyword auctions (e.g. [3] and [1]). The fields of on-line and approximation algorithms have also approached the topic. The widely noted paper by Mehta et al. [18] presents an online algorithm, with a competitive ratio $1 - \frac{1}{\epsilon}$, when the volume and sequence of queries is unknown. Mahdian et al. [13] extend this work by considering a tradeoff depending on the level of accuracy in volume predictions. We will return to this algorithm when we discuss possible future work in section 6.

There has been a considerable amount of work done in the related field of (one-off) multi-item combinatorial auctions, leading to algorithms which are practical and efficient in a wide variety of settings. (See Schrage[16] for a useful survey). These algorithms typically employ linear or integer programming (LP/IP), exploiting a very mature and efficient group of technologies. Although the characteristics of the sponsored search problem we consider make it difficult to apply the known techniques directly—for instance the frequently repeated nature of the auctions—the approach is appealing. In particular the work of Dietrich and Forrest [8], which uses column generation to determine the set of winning bids, is suggestive.

### 1.2 Paper Outline

In the remainder of this paper we will first present a simple example which motivates the need for algorithms which take advertiser budgets into account. We then present the notation, assumptions and algorithmic framework necessary for our approach. This is followed by a detailed description of the algorithm, a description of the implementation, and our preliminary computational results. Finally, we outline some future areas of follow-up research.

## 2. MOTIVATING EXAMPLE

To illustrate how critical the proper consideration of advertiser budgets might be, and how poorly a greedy algorithm might perform in the presence of these budgets, we examine the following highly simplified example. Suppose we have two queries $q_1$, and $q_2$. A single advertiser is displayed for each query, all advertisers have the same expected clickthrough rate, and the relevant bids and budgets are shown in Table 1.

**Table 1: Bids and Budgets**

| Bidder | Bid for $q_1$ | Bid for $q_2$ | Budget |
|---|---|---|---|
| $b_1$ | $C_1 + \epsilon$ | $C_1$ | $C_1$ |
| $b_2$ | $C_1$ | 0 | $C_1$ |
| $b_3$ | $C_1 - \epsilon$ | $C_1 - \epsilon$ | $2C_1$ |

As Table 2 shows, a straightforward application of GSP that displays the highest bidder is not optimal from a revenue perspective. To see this, let us assume that within the budgets' time intervals, $q_1$ appears, followed by $q_2$. Then bidder $b_1$ would pay the second bid price $C_1$ (bid by bidder $b_2$) and exhaust his budget on query $q_1$. When query $q_2$ arrives only bidder $b_3$ is now eligible, and will only pay the reserve price, which we take to be $\epsilon$. Now consider the alternative allocation that shows $b_2$ for $q_1$, producing revenue $C_1 - \epsilon$, then shows $b_1$ for query $q_2$, also producing revenue $C_1 - \epsilon$ for a total of $2C_1 - 2\epsilon$, or nearly double the revenue of the greedy allocation. Thus, a more global viewpoint, one which takes into account the keywords throughout the time period, and the budget situation for each advertiser, can lead to increased revenues. When this simple example is complicated many fold by thousands of queries, bidders and budgets, the potential for inefficiency is obvious.

**Table 2: Allocation Options**

| Allocation | Shown for $q_1$ | Shown for $q_2$ | Total Revenue |
|---|---|---|---|
| Greedy | $b_1$ | $b_3$ | $C_1 + \epsilon$ |
| Optimal | $b_2$ | $b_1$ | $2C_1 - 2\epsilon$ |

## 3. FINDING THE OPTIMAL ALLOCATION - PROBLEM DEFINITION

Let the auction marketplace consist of a set of queries $\mathcal{Q} = \{q_1, q_2, ..., q_N\}$ and bidders $\mathcal{B} = \{b_1, b_2, ..., b_M\}$. We usually use simply the index $i$ to denote query $q_i$ and the index $j$ to refer to bidder $b_j$. The *bidding state* of the marketplace at time $t$ is defined by a (sparse) matrix $A(t)$, where $A_{ij}(t)$ is the bid amount that the $j$-th bidder's is bidding on the $i$-th query $q_i$. We assume a static bidding state ($A(t) = A$) over some time-slot. While realizing that in practice this is not true due to bid management (either manually or by software), simulations suggest the effects of these types of changes are negligible. We will accommodate this dynamic aspect by frequently resolving our model as the data evolves, as is done in many other applications. It is also assumed that bids do not change in response to the allocation rule (i.e. this is not an equilibrium analysis). For each bidder $b_j$, we denote by $d_j$ the daily budget limit specified by the bidder. $d_j$ is an account level limit, i.e., it represents a spend limit across all queries for that account[1]. If a budget is not specified, we refer to this bidder as an *unbudgeted* bidder and set $d_j = \infty$.

Given a time-slot of interest, let $v_i$ be a deterministic estimate of the number of times each query $q_i$ will appear within that time slot. For each query, we define the *bidding landscape* as an ordered set of bidder indices $L_i = \{j_p : j_p \in \mathcal{B}, p = 1, ..., P_i\}$ , where the indices $j_p$ are sorted by some ranking function, and $P_i$ is the number of bidders in the landscape for query $q_i$.

In principle, we could now formulate an optimization model in which the variables corresponded to the number of times each available ad was shown with each query. However, such a model would require complex constraints and auxiliary discrete features to ensure the ordering required by the bidding landscapes for each query. We quickly abandoned such an approach in favor of a column-oriented approach which enforces the precedence explicitly. This modeling approach has a considerable history, and was quite recently used in a related model for item allocation in combinatorial auctions (see [8]). The algorithm here is quite different, since the payoffs are not deterministic and the auction is frequently repeated, among other features, but the spirit is similar.

We now define the crucial concept of a *slate* of ads corresponding to (and in fact a subset of) the bidding landscape. These slates will correspond to the columns and variables of the linear program (LP) we formulate below. Each bidding landscape $L_i$ is mapped into a set of slates $L_i^k$ , each being a unique subset of $L_i$ which can be obtained by deleting members of $L_i$ (while maintaining the ordering) and then truncating (if necessary) to $P_i^k$ (at most $P$) members. More formally, the $k$-th slate for ad $i$ includes a unique subset (of length $P_i^k$ ) of the indices of $L_i$ , and is defined as $L_i^k = \{j_l^k : j_l^k \in L_i, l \le P_i^k \le P\}$ , where $P$ is the maximum number of slots available for advertising on the page. The indices in $L_i^k$ are ordered as in $L_i$ (i.e., in order of ranking). By convention, if there are less than $P + 1$ members an additional dummy member, bidding the reserve price, may be added for the purpose of computing second-bid prices.

Two ranking (ordering) methods have been commonly used. The first and older method, sometimes known as the

[1]We could also associate budgets with other entities such as a campaign.

"Overture method" is *bid-ranking*, so that (by a slight abuse of notation):

$$A_{i1} \geq \ldots \geq A_{ij} \geq \ldots \geq A_{iP_i}$$

This scheme has now been generally superseded by expected *revenue-ranking* where the bidders on the term are ranked by product of the bid value $A_{ij}$ and a *quality score* or *clickability* value $Q_{ij}$ which is assumed to incorporate the likelihood of the ad being clicked on, based on relevance of the ad to the query, among other factors. The bidders are thus ordered so that:

$$A_{i1}Q_{i1} \geq \ldots \geq A_{ij}Q_{ij} \geq \ldots \geq A_{iP_i}Q_{iP_i}$$

Hence in any slate $k$ for query $i$ we expect the subset of ads chosen to also satisfy:

$$A_{ij_1^k}Q_{ij_1^k} \geq \ldots \geq A_{ij_p^k}Q_{ij_p^k} \geq A_{ij_{p+1}^k}Q_{ij_{p+1}^k} \geq \ldots.$$

Since the bids reflect the maximum amount that a bidder is willing to pay for a click for this query, this implies that for bidder $j_p$ to hold his position in the slate his price per click (PPC), denoted $PPC_{ij_p}^k$, should satisfy:

$$PPC_{ij_p}^k \geq A_{i,j_{p+1}^k} \cdot \frac{Q_{i,j_{p+1}^k}}{Q_{i,j_p^k}}$$

(in practice we may add some small quantity to the right hand side and treat this as an equation). Note that this implies a modified *second bid auction* where the price per click actually paid depends on the next bid, and the ratio of clickabilities. Since setting all the $Q_{ij}$ to 1 would result in the same slate and PPC as if we had used bid ranking, we shall ignore bid-ranking as a special case and consider only revenue ranking.

Let us also introduce a *click through rate* (CTR) denoted $T_{i,j_p^k,p}$ for the rate at which the ad from bidder $j_p^k$ in position $p$ in slate $k$ for query $i$ is clicked on per showing of the slate. These CTRs are estimated based on historical click data as well as other factors, and will be used both in the algorithm and the simulation described in Section 5.4.

Assuming independence of the CTRs, we can now express the expected revenue-per-search (rps) in our model for this slate and this query as the sum of the individual expected revenues per click:

$$r_{ik} = \sum_{p=1}^{P_i^k} A_{i,j_{p+1}^k} \cdot \frac{Q_{i,j_{p+1}^k}}{Q_{i,j_p^k}} \cdot T_{i,j_p^k,p} \qquad (1)$$

We now introduce the variables $x_{ik}$, which represent the number of times slate $L_i^k$ appears in the given time slot. The expected total revenue for the time-slot, over all queries, is therefore:

$$\sum_{i=1}^{N} \sum_{k} r_{ik}x_{ik} \qquad (2)$$

We represent the *total spend* for each bidder $j$ as

$$\sum_{i=1}^{N} \sum_{k} c_{ijk}x_{ik} \qquad (3)$$

where:

$$c_{ijk} = \begin{cases} A_{i,j_{p+1}^k} \cdot \frac{Q_{i,j_{p+1}^k}}{Q_{i,j_p^k}} \cdot T_{i,j_p^k,p} & 0 < p \leq P_i^k \leq P \\ 0 & \text{otherwise} \end{cases} \qquad (4)$$

is the expected cost to bidder $j$ of appearing (in position $p$) in slate $k$ for query $i$.

## 3.1 Linear Programming Formulation

We may now formally define the following linear programming (LP) problem:

*Indices*

| | |
|---|---|
| $i = 1, ..., N$ | The queries |
| $j = 1, ..., M$ | The bidders |
| $k = 1, ..., K_i$ | The slates (for query $i$) |

*Data*

| | |
|---|---|
| $d_j$ | The total budget of bidder $j$ |
| $v_i$ | Expected number of occurrences of keyword $i$ |
| $c_{ijk}$ | Expected cost to bidder $j$ if slate $k$ is shown for keyword $i$ |
| $r_{ik}$ | Expected revenue from slate $k$ for keyword $i$ |

*Variables*

$x_{ik}$ Number of times to show slate $k$ for keyword $i$

*Constraints*

(Budget)

$$\sum_{i} \sum_{k} c_{ijk}x_{ik} \leq d_j \qquad \forall j \qquad (5)$$

(Inventory)

$$\sum_{k} x_{ik} \leq v_i \qquad \forall i \qquad (6)$$

*Revenue Objective*

Maximize $\sum_i \sum_k r_{ik}x_{ik}$

## 3.2 Alternate Objective Functions

While maximizing expected overall revenue is obviously attractive from the auctioneer's point of view, it may not appear so attractive to the individual bidders. However the LP approach can accommodate a variety of alternate objective functions. One possibility is to maximize expected *efficiency* under the assumption that the bidders have expressed their true *value* for a click by the bids. While this may not always be a valid assumption, it is reasonable to view the bid as somewhat commensurate with a bidder's true value. This would correspond to an LP objective of:

*Value Objective*

Maximize $\sum_{i,k} \sum_p A_{i,j_p^k} \cdot T_{i,j_p^k,p} \cdot x_{ik}$.

Note that this objective has coefficients computed from the first prices, not second prices.

Even more simply we may decide to optimize the number of clicks obtained. This is accomplished by using an LP objective:

*Clicks Objective*

Maximize $\sum_{i,k} \sum_p T_{i,j_p^k,p} \cdot x_{ik}$.

For these particular two objectives, we could also reformulate the LP to have a polynomial number of variables and constraints, versus using the more involved column generation approach. Regardless of how the solutions are computed, it is obvious that other objectives could be formulated. If we use the column generation approach, we might take some composite weighted sum of the value, clicks, and revenue objectives. In general the "Value Objective" has advantages when considering the problem from the perspective of economics.

## 4. COLUMN GENERATION

It is clear that the number of potential columns in the above LP could be very large indeed. For each query, the number of possible slates is exponential in the number of budgeted bidders on the query, and there are many queries. It is therefore impractical to enumerate all the possible columns corresponding to the variables $x_{ik}$. If there were no binding budgets, then the optimal solution would be the trivial one consisting only of the top $P$ bidders from the landscape (called a *base slate*) being shown for each query. In fact we would expect, and experiment confirms, that for many queries, the base slate will be the only one shown even when there are active (binding) budget constraints, while some others use multiple slates—usually not more than a handful. The trick is to know which handful. The same situation occurs in many other LP applications where each column represents one of many possible complex activities. Early examples included models where the column represented a path through a network or a cutting pattern for cutting up stock sizes of material. These particular models require that small auxiliary models be solved to generate relevant columns—shortest path problems in the former case, and a "knapsack" problem in the latter.

Using a conventional column-generation approach (see [8], [12] ), we do not attempt to generate every slate $L_i^k$ a priori, but to generate an initial subset (say the $L_i$) and then generate columns as needed using the dual values of the linear program. Considering to begin with the revenue maximizing objective, let $\pi_j$ be the marginal value for bidder $j$'s budget, i.e. the simplex multipliers[6] for the $j^{th}$ constraint (5) and let $\gamma_i$ be the marginal value for the $i^{th}$ keyword, i.e. the simplex multipliers for the $i^{th}$ constraint (6), then a column corresponding to slate $L_i^k$ (and hence to variable $x_{ik}$) can be profitably introduced into the model if:

$$r_{ik} - \sum_{j \in L_i^k} c_{ijk}\pi_j - \gamma_i > 0 \qquad (7)$$

For each keyword $i$ we seek to maximize $r_{ik} - \sum_{j \in L_i^k} c_{ijk}\pi_j$ (or equivalently, minimize $\sum_{j \in L_i^k} c_{ijk}\pi_j - r_{ik}$) over all legal slates $L_i^k$. If a slate is found such that (7) is satisfied, the corresponding slate and its variable are introduced into the problem. If no such slate exists (for any $i$) then an optimal solution has been obtained.

Looking at the structure of the coefficients in (7), we see that:

$$r_{ik} = \sum_{p=1}^{P_i^k} T_{i,j_p^k,p} \cdot A_{i,j_{p+1}^k} \cdot \frac{Q_{i,j_{p+1}^k}}{Q_{i,j_p^k}} \qquad (8)$$

and

$$c_{ij_p^k k} = T_{i,j_{k_p},p} \cdot A_{i,j_{p+1}^k} \cdot \frac{Q_{i,j_{p+1}^k}}{Q_{i,j_p^k}} \qquad (9)$$

so the subproblem is to maximize (for the given $\pi_j$):

$$F_{ik}(\pi) = \sum_{p=1}^{P_i^k} T_{i,j_p^k p} \cdot A_{i,j_{p+1}^k} \cdot \frac{Q_{i,j_{p+1}^k}}{Q_{i,j_p^k}} \cdot (1 - \pi_{j_p^k}) \qquad (10)$$

over all legal slates $L_i^k$. When the number of budgeted bidders for a query is not too large this may be done by enumerating all legal subsets of $L_i^k$ , evaluating (10) on the fly, until a $L_i^k$ is found for which $F_{ik} > \gamma_i$. The corresponding column is then added to the problem. If no new column satisfying this condition is found, the present solution is optimal. If there are more than a few such legal subsets, we may need to algorithmically generate columns (slates) which maximize (10) and test whether they satisfy $F_{ik} > \gamma_i$. If the maximizing slate does not satisfy this inequality we may pass on to the next query, since no improving slate can be found for the present set of $\pi_j$ values. If this is true for all queries, then the current solution is optimal.

The overall algorithm proceeds by generating improving sets of slates in this way, then re-solving the LP, until no further improvement is possible, or some other heuristic termination criterion is met (such as percentage improvement in the objective).

The actual algorithm used to generate improving slates can be somewhat intricate, and is given in detail in a companion paper[15]. It suffices to say here that the slate generation algorithm is a form of cardinality constrained knapsack problem[7], complicated by the ordering requirement and the fact that only certain items can be omitted. The algorithm given in [15] is an efficient form of dynamic program, which given the dimensions of the problems we are considering here (perhaps a dozen ads chosen from a few dozen candidates) is fast enough to be solved many thousands of times in the column generation process.

Column generation extends to the alternate objective functions we have suggested. In particular, if the maximum value objective is chosen, we see that the objective function coefficients are:

$$\sum_p A_{i,j_p^k} \cdot T_{i,j_p^k,p} \qquad (11)$$

which must replace $r_{ik}$ in (7). The function we wish to maximize in this case is then:

$$F_{ik}(\pi) = \sum_{p=1}^{P_i^k} T_{i,j_p^k p} \cdot (A_{i,j_p^k} - A_{i,j_{p+1}^k} \cdot \frac{Q_{i,j_{p+1}^k}}{Q_{i,j_p^k}} \cdot \pi_{j_p^k}) \qquad (12)$$

Once again, algorithmic details are given in [15]. In much the same way, we may generate columns for the maximum clicks objective by replacing $A_{i,j_p^k}$ by 1 in (12).

Even though we have said that complete enumeration of the slates is impractical, it turns out to be advantageous in practice to partially enumerate them. We can generate a subset of slates for each query that include the base slate, and a subset obtained by omitting a relatively small subset of the highest ranked budgeted bidders. We do this because:

1. The most highly ranked bidders are the most likely to spend heavily, and therefore to consume their budgets most quickly.

2. A substantial initial set of slates will lead to a more realistic set of initial $\pi_j$ values after solving the first LP,

giving the column generation algorithm values closer to those of the full LP.

It turns out that we can usually obtain over 98% of the true optimal value by enumerating several quite large subsets, but this is wasteful, and much slower than using the column generation algorithm, which achieves a true optimum.

## 5. COMPUTATIONAL DETAILS

There are a number of practical details which must be considered, and overcome, to use the algorithm we have described in practice[2]. These include problem scope, solution speed and frequency, and integration with the ad serving architecture.

### 5.1 Query selection

Since there are tens of millions, perhaps hundreds of millions, of queries, we are necessarily limited to working with a subset of them. Clearly, we wish to deal with a manageable subset which captures a large part of the benefits we hope to gain from our optimization algorithm. As we have already remarked, this is aided by the typical distribution of query volumes, where the head queries capture a disproportionate share of the revenue from sponsored search. In confirmation of this, we found that in one sample, the top 5000 queries captured a significant fraction of the revenue. This indicates that even a modest gain for the head queries can lead to a significant overall gain.

### 5.2 Adjusted Bidder Budgets

This "cherry picking" can only be achieved at some cost. While it is easy to segment the queries into the head and the rest, it is not so obvious how to segment the bidders. We must somehow isolate the bidders associated with these chosen queries. Unbudgeted bidders present no problem, but we must take into account the fact that some budgeted bidders may have bid on queries in both the chosen head set, and the remainder. We must therefore partition the budgets of these bidders into two parts—that spent on our chosen set of queries, and the rest. As a practical matter, this is not too difficult; we may base the division on historical data. However, this obviously introduces a measure of uncertainty that we would wish to minimize.

Fortunately, this sort of problem has been considered before. Carrasco et al [4] consider the problem of subdividing a query-bidder bipartite graph corresponding to a sponsored search market into submarkets as sparsely connected as possible, which we may consider a generalization of our present problem of isolating a lucrative head market. Using a variant of the algorithms in [4] we may hope to choose a set of head queries minimally connected to the remainder. Having done so, we may then compute adjusted budgets for the budgeted bidders which straddle the chosen/non-chosen query set. While this a desirable property, it is not essential to our approach.

### 5.3 Column Generation Implementation

Our prototype system is implemented in C++ to run under Linux (or Cygwin) on an Intel-based work station. We use the open source COIN-OR library[11] and its LP code Clp[14], which allows efficient implementation of the column-generation framework and fast updating of the model without the need for any external interface. The column generation code itself—both the initial enumeration and subsequent dynamic programming subproblem solution—are also implemented in C++. This is economical and avoids compatibility problems, while at the same time allowing us, through the COIN-OR interface, to easily use a commercial LP code if we should ever find Clp inadequate for our models. So far this has been far from the case. The models on which we have carried out most of our experiments, using real data on approximately 5,000 queries and about 50,000 bidders of whom about 60% are budgeted, are solved to optimality in less than half a minute on a 32-bit Linux box with a 2.8 GHz Xeon processor, and in less than 1 minute, even when doubled in size. We therefore expect the algorithm to scale well, even if we were to re-solve at short intervals of, say, 15 minutes.

Note that the algorithm naturally produces results which are suitable directly for serving with organic query results, a task normally performed by the *ad server*. The ordered slates of ads, and their serving frequencies in response to queries, can be used to relieve the ad server of the need to execute the auction process for our chosen set of queries. Since we choose from the head queries, this can lead to a significant reduction in workload at the ad server itself.

### 5.4 Simulation Methodology

In order to test our approach, we measured its performance against a greedy baseline algorithm, which allocates to a query all bidders who have any remaining budget.

For this evaluation we used a fixed set of 5000 queries (see 5.1), and captured hourly (over an eleven-day period) the bidders, bids and budgets for each of the queries in this set. We used predictors of the click-through rates that use historical click data for their prediction (the exact details of these predictors is beyond the scope of this paper).

We used an impression predictor to predict $v_{it}$, the number of times query $i$ will appear at hour $t$ of the day. To compensate for the dynamics in impression volumes, we adapted our algorithm as follows: we convert the variables $x_{ik}$ to frequencies $f_{ik} = \frac{x_{ik}}{v_i}$. Then, for each query instance we use a coin-toss, weighted by the frequencies $f_{ik}$, to decide which slate to show for this query.

For each algorithm we evaluate, we then perform the following steps, every hour:

1. Simulate the keyword auction mechanism for all queries each hour in arbitrary order (the order will not matter) according to the algorithm chosen. For the greedy algorithm the auction is performed with all candidates that have any budget remaining, while for the LP based algorithms, a coin is tossed to decide which slate to show based on the frequencies $f_{ik}$.

2. Compute metrics such as revenue, efficiency, clicks and PPC.

3. Check if any bidder has exceeded their adjusted daily budget. If so, we reimburse those bidders the amount they are owed and remove them from participating in future auctions.

[2]The results we report here are based on simulation only, and the algorithm presented is not in operation on Yahoo!s production ad system.

There are several inaccuracies in the simulation. First, due to the hourly granularity, bidders who exceed their budget in the middle of a given hour may cause illegitimate price hikes for other bidders. Second, we assume at each showing of the slate that we receive exactly the expected number of clicks on each advertisement. In reality, there will be some amount of variance and inevitable inaccuracies in the click-through estimates. However, we believe that the above inaccuracies are not significant to our results; furthermore, the inaccuracies will have similar affects on both of the algorithms, so in terms of measuring relative performance these inaccuracies have little impact.

## 5.5 Results

Our simulation results are quite promising. Whether we optimize for revenue or efficiency, in both cases our results show a significant increase in both values. As we would anticipate, we get better performance in efficiency when that is the objective function, and similarly for revenue. However, the revenue appears to be more volatile in response to the objective function, with a percent increase that roughly doubles, whereas the percent increase for efficiency sees roughly a 30% increase. It is also interesting to note that the gains in revenue and efficiency for each day of the simulation, as seen in Figures 1 and 2, follow similar patterns, peaking and dipping on the same days. This suggests that the ability to maximize in either case is based on similar properties of the problem. We see that revenue and efficiency are closely tied together and that gains along one objective imply similar gains along the other.

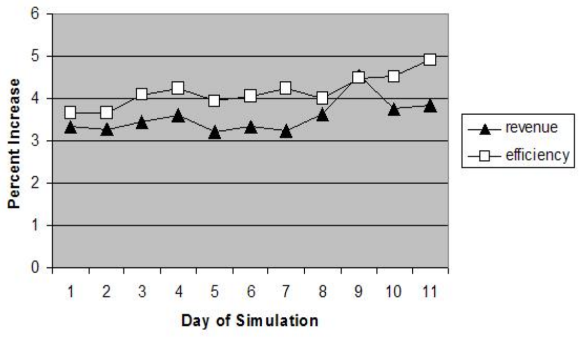

**Figure 1: Gains When Efficiency is Maximized**

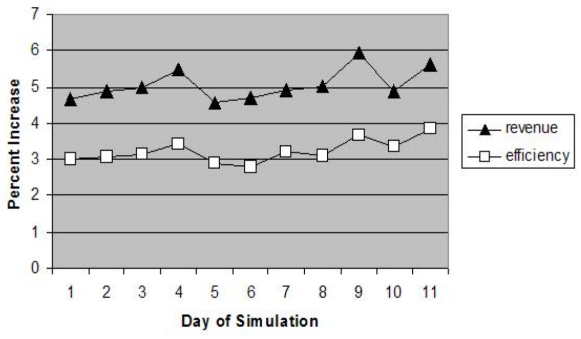

**Figure 2: Gains When Revenue is Maximized**

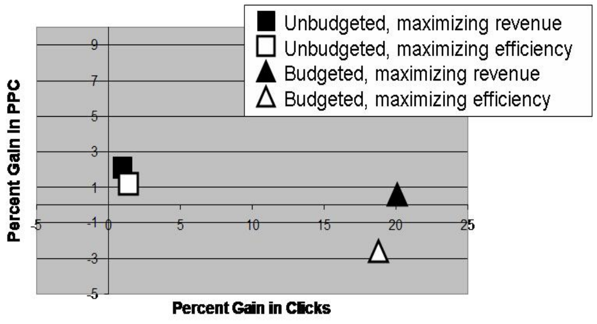

**Figure 3: Impact of the optimization on bidders**

Figure 3 shows that the impact on advertisers is favorable, but differs between budgeted and unbudgeted advertisers. We see here an interesting distinction: budgeted bidders get a steep increase in clicks with low prices, whereas the unbudgeted advertisers receive little increase in overall clicks and a slightly higher PPC (and value, although not graphed here). Overall, the impact in both cases is positive: an increase in clicks and a PPC that drops or slightly increases but remains proportional to value per click is likely to have a sustainable impact. We emphasize that our simulations do not take into account that bidders might react by changing their bids and reported budgets.

## 6. FUTURE WORK

We focus on queries at the head of the query distribution. However, there is also a heavy tail in this distribution, and many of these queries are not as easily forecasted as our subset. As described in section 1.1, when query frequencies cannot be forecasted with a high degree of certainty, previous research has proposed online algorithms with provable worst case performance guarantees. This previous research does not incorporate pricing and ranking into their solution. However, there may be a way to extend the work in [13] to use our LP as a subroutine. We are also exploring other approaches when query frequencies are unknown, including machine learning and stochastic programming.

We may also consider using parallel processing to scale up the approach even further. In particular, instead of dividing queries into our chosen set and the rest, we may use an algorithm such as that proposed in [4] to partition the query-bidder graph into multiple submarkets, and apply the budget adjustment method discussed in section 5.2 to partition the affected budgets and allow parallel solution.

It is not clear at this point how advertisers might react to our approach, and specifically how advertiser bids and reported budgets might change. Although we optimize for efficiency, this is the efficiency of the overall system, and any individual user may receive a disproportionate amount of benefit or detriment as a result of the optimization. Even if we use a truthful pricing scheme, such as VCG [19, 5, 10] or the laddered auction [2], optimizing over the overall system results in a system that may not align player incentives. An interesting area for future work is to extend our formulation to account for individual advertiser incentives.

## 7. ACKNOWLEDGEMENTS

The authors wish to thank their colleagues Andrei Z. Broder, Jan Pedersen, Michael Schwarz, Kevin Lang and S. Sathiya Keerthi for many helpful discussions, and their encouragement in the course of this research.

## 8. REFERENCES

[1] Z. Abrams. Revenue maximization when bidders have budgets. In *Proc. Symposium on Discrete Algorithms*, pages 1074–1082, 2006.

[2] G. Aggarwal, A. Goel, and R. Motwani. Truthful auctions for pricing search keywords. In *Proc. 7th ACM conference on Electronic Commerce*, pages 1–7, 2006.

[3] C. Borgs, J. Chayes, N. Immorlica, M. Mahdian, and A. Saberi. Multi-unit auctions with budget-constrained bidders. In *Proc. 6th ACM Conference on Electronic Commerce*, pages 44–51, 2005.

[4] J. J. Carrasco, D. Fain, K. Lang, and L. Zhukov. Clustering of bipartite advertiser-keyword graphs. *Workshop on Large Scale Clustering at IEEE International Conference on Data Mining*, 2003.

[5] E. Clarke. Multipart pricing of public goods. *Public Choice, 11:17-33*, 1971.

[6] G. B. Dantzig. *Linear Programming and Extensions.* Princeton University Press, Princeton, NJ, 1963.

[7] I. de Farias and G. Nemhauser. A polyhedral study of the cardinality constrained knapsack problem. *Mathematical Programming (Ser. A)*, 96:439–467, 2003.

[8] B. Dietrich and J. J. Forrest. A column generation approach for combinatorial auctions. *Workshop on Mathematics of the Internet: E-Auction and Markets Institute for Mathematics and its Applications*, 2001.

[9] B. Edelman, M. Ostrovsky, and M. Schwarz. Internet advertising and the generalizaed second price auction: Selling billions of dollars worth of keywords. *Second Workshop on Sponsored Search Auctions, Ann Arbor, MI. June*, 2006.

[10] T. Groves. Incentives in teams. *Econometrica, 41:617-631*, 1973.

[11] R. Lougee-Heimer, F. Barahona, B. L. Dietrich, J. Fasano, J. J. Forrest, R. Harder, L. Ladanyi, T. Pfender, T. Ralphs, M. Saltzman, and K. Scheinberg. The COIN-OR initiative: accelerating operations research progress through open-source software. *ORMS Today*, 28(5), 2001.

[12] M. E. Lubbecke and J. Desrosiers. Selected topics in column generation. *Operations Research*, 53(6):1006–1027, 2005.

[13] M. Mahdian, H. Nazerzadeh, and A. Saberi. Allocating online advertisement space with unreliable estimates. *ACM Conference on Electronic Commerce*, 2007.

[14] COIN-OR Foundation: `http://www.coin-or.org+`.

[15] S. Sathiya Keerthi and J. A. Tomlin. Constructing an optimal slate of advertisements. *Yahoo! Research Report*, 2006.

[16] L. Schrage. Solving multi-object auctions with LP/IP. *University of Chicago, Unpublished Manuscript*, 2001.

[17] C. Silverstein, H. Marais, M. Henzinger, and M. Moricz. Analysis of a very large web search engine query log. *SIGIR Forum*, 33(1):6–12, 1999.

[18] A. Mehta, A. Saberi, U. Vazirani, and V. Vazirani. Adwords and the generalized bipartite matching problem. In *Proceedings of the Symposium on the Foundations of Computer Science*, pages 264–273, 2005.

[19] W. Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *Journal of Finance, 16:8-37*, 1961.