Handling Forecast Errors While Bidding for Display Advertising

Kevin J. Lang
Yahoo! Research
Santa Clara, CA
langk@yahoo-inc.com

Benjamin Moseley*
University of Illinois
Urbana, IL
bmosele2@illinois.edu

Sergei Vassilvitskii
Yahoo! Research
New York, NY
sergei@yahoo-inc.com

ABSTRACT

Most of the online advertising today is sold via an auction, which requires the advertiser to respond with a valid bid within a fraction of a second. As such, most advertisers employ *bidding agents* to submit bids on their behalf. The architecture of such agents typically has (1) an offline optimization phase which incorporates the bidder's knowledge about the market and (2) an online bidding strategy which simply executes the offline strategy. The online strategy is typically highly dependent on both supply and expected price distributions, both of which are forecast using traditional machine learning methods. In this work we investigate the optimum strategy of the bidding agent when faced with incorrect forecasts. At a high level, the agent can invest resources in improving the forecasts, or can tighten the loop between successive offline optimization cycles in order to detect errors more quickly. We show analytically that the latter strategy, while simple, is extremely effective in dealing with forecast errors, and confirm this finding with experimental evaluations.

Categories and Subject Descriptors

F.2.0 [**Analysis of Algorithms and Problem Complexity**]: General; G.2.3 [**Mathematics of Computing**]: Discrete Mathematics—*Applications*

General Terms

Algorithms, Theory, Experimentation

Keywords

Ad Exchanges, Bidding Agents, Adaptive Bidding

1. INTRODUCTION

Online advertising is a multi-billion dollar industry, with billions of auctions taking place daily. At such scales automated bidding agents have become the norm. An automated agent takes as input a set of parameters, or goals, of the advertiser: for example, the targeting constraints, desired number of impressions, quality constraints, etc., and then outputs a bid for every advertising opportunity. The bids from all interested parties are collected by the publisher, often through an advertising exchange (e.g. RightMedia Exchange, Google Ad Exchange), and the opportunity to show an advertisement is given to the highest bidder.

The goals of the bidding agents are quite diverse: some bidding agents aim to maximize the number of clicks or conversions for an advertiser, others look to achieve a representative allocation, aiming for a uniform slice of all eligible impressions [8], still others care about temporal smoothness, making sure the advertiser receives a stream of impressions throughout the day, as opposed to getting them all in the morning or in the evening. Regardless of the specific goal, almost all of the agents rely on some sort of forecasting in order to properly set the bidding parameters. Specifically, almost all of the optimization formulations rely on *supply* and *price* forecasts. Intuitively, the forecasts are needed to judge the utility of a specific impression. Deciding whether an impression looks cheap or expensive requires the knowledge of the typical price of such impressions. Similarly, knowing how many of such impressions will arrive in the future helps decide the urgency with which the bidding agent should be bidding, since most advertisers have an additional budget constraint on their campaigns.

Accurately predicting both supply and price forecasts is a non-trivial endeavor. For example, for the supply forecasts, although general traffic trends stay consistent day to day—there are more impressions in the middle of the day than in the middle of the night—the exact forecasts experience a lot of daily fluctuations —the Oscars, for example will change the traffic pattern for websites related to the entertainment industry.

Moreover, an individual bidder may not have access to the necessary data to make the desired predictions. Some exchanges employ selective call out strategies [1], wherein the number of eligible impressions observed by the bidder is correlated with his win percentage. In other situations, when the bidding agents are hosted by the exchange, the exchange only reports the number of *won* opportunities for each bidder, not the number of *eligible* opportunities. (In fact early termination methods [11] result in situations where a full eligible set is *never* computed for each opportunity.)

All of the above leads to a natural question for a bidding agent designer: how to deal with forecast errors in developing a bidding strategy. One approach requires investing heavily in better forecasting strategies to reduce the number of errors, but one quickly hits a state of diminishing returns: there is enough entropy in the system, so that perfect fore-

*Work done while visiting Yahoo! Labs. Partially supported by the NSF grant CCF-1016684.

casts are simply not achievable. A different approach is to myopically adjust the bidding parameters based on the past history, relying on frequent re-optimization to achieve the goals of the agent. In this work we formally analyze the latter approach, and show that as long as the adjustments are done sufficiently frequently, the agent can automatically compensate for reasonably large errors in forecasts.

1.1 Related Work

Advertising exchanges sell billions of impressions per day requiring the use of automated bidding agents to buy advertising opportunities online [13]. The typical problem facing a bidding agent is to obtain a target number of impressions to users satisfying specific targeting characteristics (e.g. Male, Age > 30) over some limited time horizon (e.g. one week), while limited by a budget constraint.

The problem of finding a good bidding strategy falls under the larger umbrella of allocation problems for display advertising. Previous work focused on the allocation problem in the Guaranteed Advertising scenario, where a publisher is trying to satisfy multiple display advertising contracts simultaneously, by deciding which ad to show to each user visiting his page. This problem is often modeled as an online matching problem, which has a rich history beginning with the seminal work by Karp et al. [10] who showed a lower bound of $1 - {}^1\!/_e$ on the competitive ratio of any algorithm, and gave a randomized algorithm that matched that bound. This work has been applied to the online advertising scenario in a series of works by Feldman et al. [6, 7], who used the specifics of the problem to improve on that bound.

A different direction was to phrase the problem as a stochastic optimization problem, where the user arrivals are not adversarial, but rather are drawn from some distribution. Devanur and Hayes [5] assumed that the arrivals are independent and effectively showed how to learn the distribution before performing the analysis. Vee et al. [14], assumed full knowledge of the distributions and focused on providing a compact strategy that can be readily implemented in ad serving. Only the recent work by Chen et al. [3] explicitly addressed the limitations of supply forecasts and experimentally showed that control theory based methods mitigate the impact of forecast errors. In this work we show both analytically and experimentally that even simpler methods lead to good performance.

The allocation problems described above focus on obtaining the target number of impressions in situations where the publisher controls the allocation, and hence no explicit bidding is necessary. Requiring that the bidding agent buy the impressions directly from an exchange adds an additional layer of complexity. In addition to supply forecasts, the competitors bids need to be forecast as well [4], or learned in real time [9]. Moreover, the choice of the objective function for the bidder becomes more important. For example, Ghosh et al. [8] conclude that one should aim for a *fair* or *representative* allocation, rather than getting the cheapest impressions possible.

1.2 Our Contribution

We explicitly study the problem faced by the bidding agent designer in the face of inaccurate forecasts. We begin by analytically bounding the error in the number of impressions won and the total spend when given inaccurate forecasts and using a well known bidding strategy. We then show that a simple algorithm that myopically re-optimizes the bidding parameters can greatly mitigate the errors in the forecasts. The algorithm is simple and requires minimal feedback from the system, yet a formal analysis shows that the re-optimization approach quickly converges to the bidder's desired budget and demand. We prove the algorithm's effectiveness as a function of the number of update cycles and the original error in the forecasts. We conclude with an experimental analysis on both synthetic and real-world datasets. Our experimental evaluations shows that the algorithm performs extremely well on real data, even with forecasts with large error.

2. PRELIMINARIES

We consider the problem from the perspective of a single advertiser, Alice, bidding in a second price auction. Alice has a total budget of $\mathcal{B}$ and desires to win $\mathcal{D}$ impressions. We note that Alice's goal is *not* to win $\mathcal{D}$ impressions at minimum cost: as [8] argued, advertisers in these markets have a common value, which means the cheapest impressions are precisely the lowest quality ones. This setting best describes advertisers buying individual impressions.[1] At every opportunity, Alice can submit a bid b. Other advertisers submit bids as well, and we model the highest competing bid as being chosen independently from a distribution with density p^*, and CDF of P^*. Let $c \sim P^*$ be the highest competitor bid. If $b > c$ then Alice wins the impression, decrementing her desired demand by 1 and her budget by c. Otherwise, $c \geq b$ and Alice loses the impression, leaving her demand and her remaining budget unchanged.

If the distribution P^* and the total number of impressions n^* is known to Alice, she can use an easy bidding strategy that achieves her demand while spending exactly the budget. We call this algorithm `SingleRound`. To analyze the algorithm, we will denote by $D_{\mathcal{A}}$ the total number of auctions won using bidding strategy $\mathcal{A}$, dropping the subscript when it's clear from the context. Similarly, denote by $B_{\mathcal{A}}$ the expected budget spent using strategy $\mathcal{A}$.

Let $t = \mathcal{B}/\mathcal{D}$ be the desired *target* spend per win. If $t \leq \mathbb{E}[P^*]$ then there exists a bid b^* so that the expected price to Alice conditioned on winning the impression is exactly t, i.e.

$$E[p|p < b^*] = t.$$

The expected number of impressions won by bidding b^* is exactly $n \cdot P^*(b^*)$. The `SingleRound` bidding strategy bids b^* with probability $q^* = \frac{\mathcal{D}}{n^* \cdot P^*(b^*)}$ (For simplicity we assume that $q^* \leq 1$, we investigate this further in Section 5.) It is easy to see that $\mathbb{E}[D] = \mathcal{D}$ and $\mathbb{E}[B] = \mathcal{B}$.

Unfortunately, in practice, neither the total supply, n^*, nor the price distribution, P^* are known ahead of time, instead only forecasts, which we denote by n and P respectively, are available. To quantify the forecast error, we will use δ to denote the relative error in the supply forecast: $n^*(1 - \delta) \leq n \leq n^*(1 + \delta)$. Similarly, we denote by γ the relative error in the price forecast: for any $b \geq 0$, $p^*(b)(1-\gamma) \leq p(b) \leq p^*(b)(1+\gamma)$. We emphasize that while we use δ and γ for the analysis, neither of the parameters is known to the advertiser.

[1]The underlying problem is similar for CPC advertisers who only pay per click.

3. NON-ADAPTIVE ALGORITHM

In this Section we analyze the performance of a non-adaptive bidding algorithm as a function of the forecast errors, δ and γ. As we saw before, given the bidding agent parameters: the demand $\mathcal{D}$ and the budget $\mathcal{B}$, and the forecast of others' behavior: the total supply, n and the distribution of the highest competing bid, P, the algorithm SingleRound achieves the goals in expectation. Here SingleRound uses the forecasts P and n as if they were the actual supply and bid landscape to determine the bid and probability of bidding.

Now consider the performance of SingleRound in the face of forecast errors. We bound the demand fulfilled, i.e. the number of auctions won, and the budget spent:

THEOREM 3.1. *Consider the bidding strategy SingleRound, under supply forecast error, δ, and bid forecast error γ. Then*

$$\frac{\mathcal{D}}{(1+\gamma)(1+\delta)} \le \mathbb{E}[D] \le \frac{\mathcal{D}}{(1-\gamma)(1-\delta)},$$

and

$$\frac{\mathcal{B}}{(1+\gamma)(1+\delta)} \le \mathbb{E}[B] \le \frac{\mathcal{B}}{(1-\gamma)(1-\delta)}.$$

PROOF. Let y be the bid computed by SingleRound, and q the probability of participating. Then the expected number of impressions won is

$$\begin{aligned}\mathbb{E}[D] &= q \cdot P^*(y) \cdot n^* \\ &= \frac{\mathcal{D}}{nP(y)} \cdot P^*(y) \cdot n^* \\ &\in \left[\frac{\mathcal{D}}{(1+\delta)(1+\gamma)}, \frac{\mathcal{D}}{(1-\delta)(1-\gamma)}\right]\end{aligned}$$

To bound the total budget spent, consider the expected amount spent:

$$\begin{aligned}\mathbb{E}[B] &= n^* \cdot q \cdot P^*(y) \cdot \mathbb{E}_{P^*}[x : x < y] \\ &= n^* \cdot \frac{\mathcal{D}}{n \cdot P(y)} \cdot P^*(y) \cdot \mathbb{E}_{P^*}[x : x < y] \\ &= \mathcal{B} \cdot \frac{n^*}{n} \cdot \frac{P^*(y)}{P(y)} \cdot \frac{\mathbb{E}_{P^*}[x : x < y]}{\mathbb{E}_P[x : x < y]}\end{aligned}$$

Since $\mathbb{E}_P[x : x < y] = \frac{1}{P(y)} \int_{b=0}^{y} bp(b)db$, we can simplify to:

$$\begin{aligned}\mathbb{E}[B] &= \mathcal{B} \cdot \frac{n^*}{n} \frac{\int_{b=0}^{y} bp^*(b)db}{\int_{b=0}^{y} bp(b)db} \\ &\in \left[\frac{\mathcal{B}}{(1+\delta)(1+\gamma)}, \frac{\mathcal{B}}{(1-\delta)(1-\gamma)}\right],\end{aligned}$$

where the last line follows because $p^*(b) \in \frac{p(b)}{1\pm\gamma}$ for all b.

□

At this point we have shown that a simple algorithm will have bounded error in the expected budget spent and the expected number of auctions won. However, this error maybe quite large depending on the exact values of δ and γ. In the next section we show that a myopic adaptive algorithm can dramatically reduce the overall error.

4. ADAPTIVE ALGORITHM

We show that the simplest adaptive strategy, which periodically reruns the SingleRound Algorithm using updated budget and impressions targets, can be very powerful in limiting the effect of the forecast errors. The only intermediate information required to rerun SingleRound is the total amount spent and total number of impressions won during the previous time period. We state our results more broadly for any optimization algorithm $\mathcal{A}$ that takes a supply forecast, bid landscape forecast, desired demand and remaining budget.

4.1 Analysis

For the purposes of the analysis, we assume that impressions begin arriving at time $t = 0$ and denote by T the expiration date of the contract. Let $k \in \mathbb{N}^+$ denote the number of optimization routines performed in time T. We will assume that supply is distributed uniformly during the T time steps. This assumption does not hold in practice, but makes for a simpler analysis. We investigate its effect on the algorithm's performance in Section 5. If n^* is the total available supply, then $n^* \cdot (1 - \frac{i}{k})$ opportunities are available after time $T \cdot i/k$, which will also serve as the time of the $i+1$-st optimization by $\mathcal{A}$.

Our goal is to bound the expected budget spent and the expected demand received by the optimization algorithm as a function of the forecast errors δ and γ and the number of optimization cycles, k. We will show that as k increases, the total error in the demand received and budget spent decreases as a function of k. To do this, we abstract both problems into one general problem. In this problem there are n^* events. During each event the algorithm makes a decision and experiences some reward. Let Y_i be the random variable denoting the reward during the ith event. Let $X = \sum_{i=1}^{n^*} Y_i$. The goal of the algorithm is to make decisions on each event so that the expected value of X is equal to a goal parameter, α. In the case of analyzing the demand, $\alpha = \mathcal{D}$ and Y_i is the probability of winning the impression. In case of analyzing the total spend, $\alpha = \mathcal{B}$ and Y_i is the expected spend per impression.

The goal of this section is to prove the following theorem, which roughly shows that the error experienced by the algorithm drops as $1/k$, and so even a small number of re-optimization cycles can go a long way towards improving performance. (We note that a more specialized version of this theorem geared specifically for performance of target demand appears in [2].)

THEOREM 4.1. *Let $\epsilon, \epsilon' > 0$. If $\mathcal{A}$ ensures that*

$$(1-\epsilon')\frac{\alpha}{n^*} \le \mathbb{E}[Y_i] \le (1+\epsilon)\frac{\alpha}{n^*}$$

for any possible value of n^*, α and $i \in [n^*]$ then re-running $\mathcal{A}$ $k \ge 1$ times every n^*/k events, guarantees:

$$-\alpha\epsilon' \left(\frac{1}{k}\right)^{1-\epsilon'} \le \mathbb{E}[X - \alpha] \le \alpha\epsilon \left(\frac{1}{k}\right)^{1-\epsilon'}.$$

The algorithm $\mathcal{A}$'s goal is to have $\mathbb{E}[X] = \alpha$, however the algorithm may not be able to achieve this in expectation. This is because the algorithm may have some bounded error as described by ϵ and ϵ'. Now, if we use the algorithm to optimize k times, then the algorithm converges on $\mathbb{E}[X] = \alpha$

at a rate of $\left(\frac{1}{k}\right)^{1-\epsilon'}$. Thus, simply optimizing k times makes the algorithm converge on the correct value quickly. For a concrete example, let $\mathcal{A}$ be `SingleRound`, $\epsilon = \frac{\delta+\gamma-\delta\gamma}{(1-\delta)(1-\gamma)}$, $\epsilon' = \frac{\delta+\gamma+\delta\gamma}{(1+\delta)(1+\gamma)}$ and α to be either the budget or the demand. In this case, using previous theorem with Theorem 3.1 we have the following corollary.

Corollary 4.2. *If the SingleRound is rerun k times at even time intervals between 0 and T, then:*

$$(1-\epsilon'\left(\frac{1}{k}\right)^{1-\epsilon'})\mathcal{D} \le \mathbb{E}[D] \le (1+\epsilon\left(\frac{1}{k}\right)^{1-\epsilon'})\mathcal{D}$$

and

$$(1-\epsilon'\left(\frac{1}{k}\right)^{1-\epsilon'})\mathcal{B} \le \mathbb{E}[B] \le (1+\epsilon\left(\frac{1}{k}\right)^{1-\epsilon'})\mathcal{B}$$

As an example, suppose that both supply and price forecasts are off by 25%: $\delta = \gamma = 1/4$. Then, a single optimization may lead to an underdelivery factor of $36\% = \epsilon'$, or a budget overspend of $78\% = \epsilon$. A single additional retopimization (setting $k = 2$), reduces both by a factor of $\frac{1}{2}^{1-\epsilon'} \approx 0.78$, or over 20%. Having $k = 10$ re-optimizations, leads to a reduction in error of more than half ($\approx 56\%$).

To show the previous theorem, let $X_i = \sum_{j=1}^{n^* i/k} Y_i$ be the expected value given to the algorithm just before the ith optimization where $1 \le i \le k+1$. (X_{k+1} denotes the value remaining in the end.) Define the recurrence $R(i) = \alpha - \mathbb{E}[X_i]$. Intuitively, $R(i)$ is either the *remaining* demand or budget just before the ith optimization. Note that $R(1) := \alpha$. Further, $R(i+1) = \alpha - \mathbb{E}[X_{i+1}] = \alpha - \mathbb{E}[X_i] - (\mathbb{E}[X_{i+1}] - \mathbb{E}[X_i]) = R(i) - (\mathbb{E}[X_{i+1}] - \mathbb{E}[X_i])$. The goal is to find a lower bound and upper bound on $R(k+1)$. To do this, the proof proceeds as follows. First, we bound $R(i)$ in terms of $R(i-1)$. Using this we can upper bound $R(k)$. With an upper bound on $R(k)$ we can derive the final bounds on $R(k+1)$.

Lemma 4.3. $(1 - \frac{1+\epsilon}{k-i+1})\mathbb{E}[R(i)] \le \mathbb{E}[R(i+1)] \le (1 - \frac{1-\epsilon'}{k-i+1})\mathbb{E}[R(i)]$ *for any $1 < i \le k$.*

Proof. Consider the time the ith optimization is performed. At this time there are $n^*(1-\frac{i-1}{k})$ events remaining. There will be n^*/k events available before the $(i+1)$st optimization. Therefore a $1/(k-i+1)$ fraction of the remaining events occur in the ith stage. We have that,

$$\begin{aligned}
\mathbb{E}[R(i+1)] &= \sum_x \mathbb{E}[R(i+1)|R(i)] \cdot \mathbf{Pr}[R(i)=x] \\
&\ge \sum_x (1 - \frac{1+\epsilon}{k-i+1})R(i) \cdot \mathbf{Pr}[R(i)=x] \\
&\qquad \text{[Definition of } \mathcal{A} \text{ and } R] \\
&= (1 - \frac{1+\epsilon}{k-i+1})\mathbb{E}[R(i)]
\end{aligned}$$

and similarly we have,

$$\begin{aligned}
\mathbb{E}[R(i+1)] &= \sum_x \mathbb{E}[R(i+1)|R(i)] \cdot \mathbf{Pr}[R(i)=x] \\
&\le \sum_x (1 - \frac{1-\epsilon'}{k-i+1})R(i) \cdot \mathbf{Pr}[R(i)=x] \\
&\qquad \text{[Definition of } \mathcal{A} \text{ and } R] \\
&= (1 - \frac{1-\epsilon'}{k-i+1})\mathbb{E}[R(i)]
\end{aligned}$$

□

Using the previous lemma we upper bound $\mathbb{E}[R(k)]$.

Lemma 4.4. *We have that $\mathbb{E}[R(k)] \le \alpha\left(\frac{1}{k}\right)^{(1-\epsilon')}$.*

Proof.

$$\begin{aligned}
\mathbb{E}[R(k)] &\le \alpha \prod_{j=2}^{k} (1 - \frac{1-\epsilon'}{k-j+2}) \\
&\qquad \text{[Lemma 4.3 and } R(1) = \alpha] \\
&\le \alpha \prod_{j=2}^{k} \exp\left(-\frac{1-\epsilon'}{k-j+2}\right) \\
&\le \alpha \exp\left(-(1-\epsilon')\sum_{j=2}^{k} \frac{1}{k-j+2}\right) \\
&\le \alpha \exp\left(-(1-\epsilon')\log k\right) \\
&\le \alpha \left(\frac{1}{k}\right)^{(1-\epsilon')}
\end{aligned}$$

□

Now we find upper and lower bounds on $\mathbb{E}[R(k+1)]$.

Lemma 4.5. $-\alpha\epsilon\left(\frac{1}{k}\right)^{(1-\epsilon')} \le \mathbb{E}[R(k+1)] \le -\alpha\epsilon'\left(\frac{1}{k}\right)^{(1-\epsilon')}$

Proof. We have that,

$$\begin{aligned}
\mathbb{E}[R(k+1)] &\ge -\epsilon\mathbb{E}[R(k)] \quad \text{[Lemma 4.3]} \\
&\ge -\alpha\epsilon\left(\frac{1}{k}\right)^{(1-\epsilon')} \quad [\epsilon > 0 \text{ and Lemma 4.4}]
\end{aligned}$$

and

$$\begin{aligned}
\mathbb{E}[R(k+1)] &\le \epsilon'\mathbb{E}[R(k)] \quad \text{[Lemma 4.3]} \\
&\le \alpha\epsilon'\left(\frac{1}{k}\right)^{(1-\epsilon')} \quad [\epsilon' > 0 \text{ and Lemma 4.4}]
\end{aligned}$$

□

Lemma 4.5 and the definition of R gives Theorem 4.1.

5. OVERVIEW OF EXPERIMENTS

The previous sections described a bidding algorithm that is tolerant of errors in the supply and price forecasts, and presented a theoretical analysis showing that, under certain simplifying assumptions, an advertiser using this bidding algorithm can nearly hit a campaign's demand target and spending target, with errors that become smaller when the campaign's lifetime is divided into more time blocks (thus causing more re-optimizations to occur).

Because the problem addressed by this paper is real, that theoretical analysis is only the first step. It is also important to find out whether the simplifying assumptions made during the analysis are reasonable, and whether the algorithm is sufficiently practical and robust to work under realistic conditions. Among the simplifying assumptions were the following:

- The analysis assumes that the supply which a campaign encounters during its lifetime is divided equally between the time intervals. In reality, the amount of supply in the various time intervals can be highly unequal.
- The analysis makes a feasibility assumption that $\mathcal{D}$ impressions can be obtained while spending $\mathcal{B}$. In the experiments, we will examine the assumption-violating case where the bid z leading to the correct demand is greater than the bid y leading to the correct spend.
- The analysis assumes that the forecasts are being used as-is, with no updates. This is a reasonable assumption given that it would be expensive to obtain the feedback necessary to update the forecasts. Nevertheless, it is natural to ask whether better performance could be obtained if the forecasts could be updated.

The next couple of sections contain experiments that address the following 5 questions:

1 Do the errors decrease with increasing k?

2 Is the algorithm robust enough to still work when the assumption about equal time intervals is violated?

3 Is the algorithm robust enough to work with real price distributions.?

4 What should the algorithm do when the assumption that $z \leq y$ is violated?

5 Can updating the forecasts yield better results?

Since there is value both in using real data, and in using non-proprietary data, we will present two sets of experiments in the next two sections. In Section 6 we will use real data to address questions 1,2,3 from the list above. In Section 7 we will use synthetic (and hence non-proprietary) data to address questions 1,2,4,5 from the list above.

6. EXPERIMENTS ON RMX DATA

In this section we give an experimental study of the algorithm introduced in this paper. The goal of this section is to show that the algorithm does not only perform well theoretically, but the algorithm, in fact, performs well on real world data sets and is robust. All of the data used in the experiments was gathered from live auctions from the RightMedia exchange. RightMedia is currently the largest ad exchange currently in industry with over nine billion transactions daily [13].

The highest bid from 100,000 auctions was collected from four separate days. We will refer to these as four separate datasets. These bids were used to represent the bid landscape. To construct the bid forecast we used the bid distribution from one of the datasets. One of the goals of the experiments was to show the robustness of the algorithm. To this end, we constructed the bid forecast on only 10,000 auctions from one of the datasets. Then we ran the algorithm, using this forecast, on the 100,000 auctions from the other three datasets, separately. For each experiment we ran the algorithm 5 times and took the average of the outputs considered (e.g. the budget spent or demand received).

Convergence of Budget and Demand with Supply and Landscape Error:.

Our first experiment is designed to show that the budget spent and demand received converges to the desired budget $\mathcal{B}$ and demand $\mathcal{D}$ as the number of optimizations increase. First we focus on the case which we call the 'non-varying interval' case. In this case, when the number of optimizations is k, the algorithm is optimized after each $1/k$ fraction of the total number of opportunities have occurred. To exemplify that the algorithm is robust, we set two parameters δ and β. The value of δ represents the error in the supply forecast. Here the forecasted supply which was input to the algorithm is $(1+\delta)$ multiplied by the actual supply. For this experiment δ was set to $1/2$ and therefore the forecasted supply for each of the experiments was $150,000$ while the actual supply was $100,000$. In practice, this would be a quite large margin of error in the supply forecast. Now consider the bid landscape forecast. In the experiment, a small sample of real bids were used to construct the supply forecast. It would seem somewhat surprising that using this as the forecast would accurately forecast bids from another day's auctions. However, the following data shows that the optimization algorithm overcomes this. To further show the robustness of the algorithm, we also used a parameter β, which is the error in the bid landscape forecast.[2] When using this parameter, the real bids used to construct the forecast were multiplied by $(1-\beta)$. In the following experiment we set $\beta = .5$. Thus, each bid in the training data set used for the bid landscape forecast was scaled by a factor of .5. Still the optimization algorithm performed well.

The second experiment performed we call the 'varying interval' case. In this case, the number of opportunities between each optimization is random. Here we choose $k-1$ random numbers between 0 and the total supply $100,000$. When the number of opportunities which occurred previously was equal to the one of the random numbers then an optimization was performed. In practice, the number of auctions which occur during a time period do vary and here we essentially consider the worst cast scenario.

For the advertiser we set the demand $\mathcal{D}$ to be $10,000$, one tenth of the total supply. The advertiser's budget was set at about 2.3% of the total sum of the bids. Thus, the advertiser desires a large fraction of the total supply (10%) while using a relatively small budget (2.3%). When bidding in the auction, if there was a tie between the advertiser and the external bid then it was assumed that the external bidder won the auction. We used the advertiser with these parameters on the auctions from the three different datasets using a different number of optimizations during each run. The data from each of the three datasets was quite similar and the graph in Figure 1 and Figure 2 shows one of the datasets.

First consider the non-varying interval case. Figure 1 shows that if the algorithm performs only a single optimization, then an average of 6349.8 auctions are won, which is about 63.5% of the desired demand. However, if the algorithm were to optimize 15 times then the demand received would increase to about 9000, within 10% of the desired demand. This shows that with a small number of optimizations the algorithm converges to the desired demand. Fur-

[2]This β controls the error in the values of the bids, and so is slightly different from the γ in the theoretical analysis, which bounds the error in the probabilities of bids.

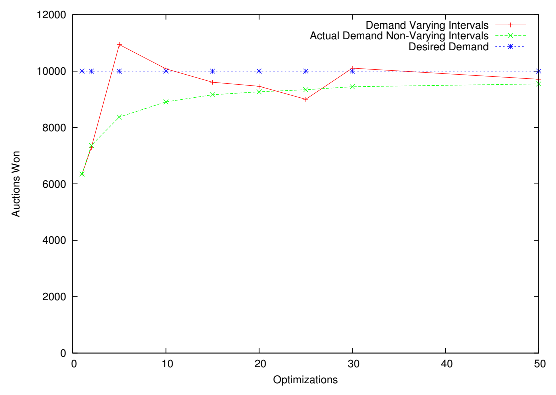


Figure 1: Demand received, as a function of number of optimizations. The desired demand was 10,000. The 'Actual Demand' is the demand received in the experiments.

ther, if the number of optimizations is as large as 50 then the algorithm is within 5% of the desired demand. Being able to optimize frequently can be difficult to do in practice because it is time consuming to receive feedback from the system. However, even if the optimizations can not be done often, the algorithm still converges well.

Figure 2 shows the corresponding budget. It can be seen that the budget has 3% error when no optimizations were performed. With a large number of optimizations the budget converges with less than .4% error with 50 optimizations. Thus, converging on the desired demand comes at little relative expense in the budget spent.

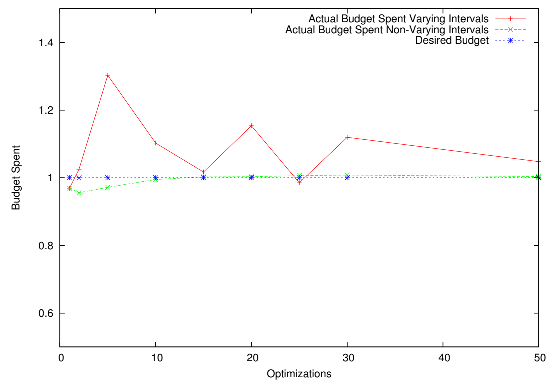


Figure 2: Budget spent, as a function of number of optimizations. The desired budget was 1.

Now consider the varying interval case. It can be seen that the demand received and the budget spent is more variable in this case. The demand received converges to within 10% of the desired demand when the number of optimizations is larger than 10. The budget is more variable, however, once the number of optimizations is larger than 10 the over spent budget was no larger than 10%. This shows that the algorithm is fairly robust to there being fluctuations in the number of opportunities available during a single optimization.

Similar results were obtained as above when the parameters δ and β were varied up to .8. However, once the error parameters became too large, the algorithm could not overcome the error in the forecasts. Further, similar results were obtain when the budget and demand were varied. However, once the demand becomes too close to the total number of opportunities, there were not enough remaining opportunities for the algorithm to converge on the budget and demand. Also when the budget was quite small the algorithm, naturally, was unable to win enough auctions.

In summary, this section contained experiments using real price distributions which yielded the answer "yes" for questions 1-3 in Section 5.

	Forecasted Supply		
Fcast. Prices	too high	accurate	too low
too low	δ=0.4 β= 0.4	δ=0.0 β= 0.4	δ= -0.2 β= 0.4
accurate	δ=0.4 β= 0.0	δ=0.0 β= 0.0	δ= -0.2 β= 0.0
too high	δ=0.4 β= -0.6	δ=0.0 β= -0.6	δ= -0.2 β= -0.6

Figure 3: The 9 scenarios tested in Section 7.

7. EXPERIMENTS ON SYNTHETIC DATA

This section describes some additional experiments that are easier to replicate because they use synthetic rather than proprietary data. First, in Section 7.1, the experimental setup is described, and some basic results are presented that once again demonstrate that the errors in hitting the campaign's demand and spending targets decrease with increasing k, and also that the algorithm is robust to unequal subdivision of supply between time intervals. Then in Sections 7.2 and 7.3, the previously unaddressed questions 4 and 5 are empirically investigated.

7.1 Experimental Setup and Basic Results

- True supply $N = 100000$ auctions.
- Forecasted supply $N_f = N \cdot (1 + \delta)$, where $(1 + \delta)$ is the simulation's forecast error factor.
- Demand Target $D = 10000$ auction wins.
- Budget Target $B = 2500$ units of money.
- Implied target spend per win = 0.25.
- Competitors' bids are drawn from the lognormal distribution exp(gaussian(mean=0,sigma=4/3)).
- The price forecast given to the bidding agent is represented by a separate ensemble of samples drawn from the same lognormal distribution, but then multiplied by the simulation's price error factor $(1 - \beta)$.
- Simulations were run with two different schemes for subdividing the supply amongst the time intervals: nearly equal random subdivision, which is probably better

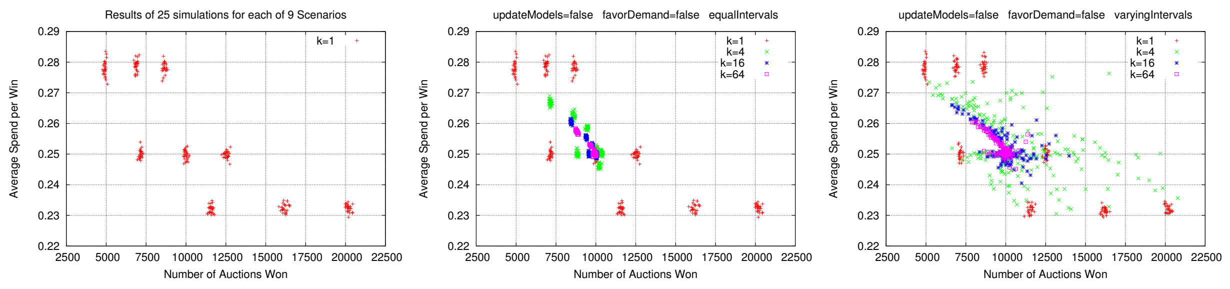


Figure 4: Left: Simulation results for k=1. The 9 obvious clusters correspond to the 9 scenarios listed in Table 3, with the same layout. Middle: Results for k in {1,4,16,64}. Notice that as k is increased, the distribution of outcomes contracts towards the target demand of 10000 and target price of 0.25. Right: the results are more scattered, but the same kind of contraction is evident when the supply is subdivided arbitrarily rather than equally between intervals.

```
Campaign Inputs: $(D, B, k)$
Model Inputs: $(N_f, P_f)$
Control Inputs: (favorDemand, updateModels)

SM $\leftarrow$ initSM($N_f, k$)      // supply model
PM $\leftarrow$ initPM($P_f$)         // price model
$D_a \leftarrow 0$; $B_a \leftarrow 0$
for $k_a = 0$ to $k-1$ do // loop over time blocks
    $D_r \leftarrow D - D_a$; $B_r \leftarrow B - B_a$; $k_r \leftarrow k - k_a$
    $N_e \leftarrow$ estimateRemainingSupply(SM, $k_r$)
    $P_e \leftarrow$ estimatePriceDistribution(PM)
    define: $F_e(y) = \frac{1}{P_e(y)} \int_0^y b \cdot p_e(b) db$
    $winrate_z \leftarrow D_r/N_e$    // aim for $D$
    $spendrate_y \leftarrow B_r/D_r$    // aim for $B$
    $bid_z = P_e^{-1}(winrate_z)$
    $bid_y = F_e^{-1}(spendrate_y)$
    $winrate_y = P_e(bid_y)$
    if ($bid_z \leq bid_y$) then    // "feasible" case
        curBid $\leftarrow bid_y$;
        bidProb $\leftarrow winrate_z/winrate_y$
    else    // "infeasible" case where ($bid_z > bid_y$)
        if (favorDemand) then curBid $\leftarrow bid_z$
        else curBid $\leftarrow bid_y$ end if
        bidProb $\leftarrow$ 1.0
    end if
    (numWins, amtSpent, numAuctions, competingBids)
        $\leftarrow$ ResultsOfBlock (curBid, bidProb)
    $D_a \leftarrow D_a$ + numWins
    $B_a \leftarrow B_a$ + amtSpent
    if updateModels then
        updateSupplyModel (SM, numAuctions)
        updatePriceModel (PM, competingBids)
    end if
end for

Outputs: $(D_a, B_a)$
```

Figure 5: Pseudocode for bidding algorithm.

than reality, and unconstrained random subdivision, as described in section 6, which is probably worse than reality.

- Nine different error scenarios were tested, each specified by values for the supply error δ and the price error β. These scenarios are listed in Table 3.
- 25 simulations were performed for every combination of (intervalScheme, ErrorScenario, k).
- The results are presented as scatter plots, with number of auctions won on the x axis, and average spend per win on the y axis.
- Pseudocode for the simulated bidding agent appears in Figure 5. There are control flags called *favorDemand* and *updateModels*. These will be explained in Sections 7.2 and 7.3, but were both set to false for the basic simulations whose results we will now discuss.

Consider the plots in Figure 4. The leftmost pane shows results for k=1 and equalIntervals. There are nine obvious clusters of points which correspond to the nine error scenarios, and in fact have the same spatial layout as the table in Figure 3. We will mention a couple of the clusters. The middle cluster in this leftmost plot is for the accurate-forecast scenario ($\delta = 0, \beta = 0$). Naturally, this cluster of outcomes is centered on the target demand of 10000 and the target price of 0.25. The upper left cluster is for the scenario ($\delta = 0.4, \beta = 0.4$). The inaccurate forecasts are causing the bidding agent to under-deliver (5000 wins) and over-spend per win (price of about 0.28).

Now consider the center pane of Figure 4, in which k ranges over {1,4,16,64}. Evidently the ensemble of outcomes (comprising 25 runs each for 9 scenarios) is contracting towards the target demand and target price as k increases. This re-affirms the earlier answer of "yes" for question 1.

Now consider the rightmost pane of Figure 4, which differs from the center pane in that the simulator subvided the supply unequally rather than equally between the time intervals. The results for the unequal subdivision are much more more scattered, but still there is an overall pattern of contraction towards the target demand and target price as k increases. This re-affirms the earlier answer of "yes" for question 2.

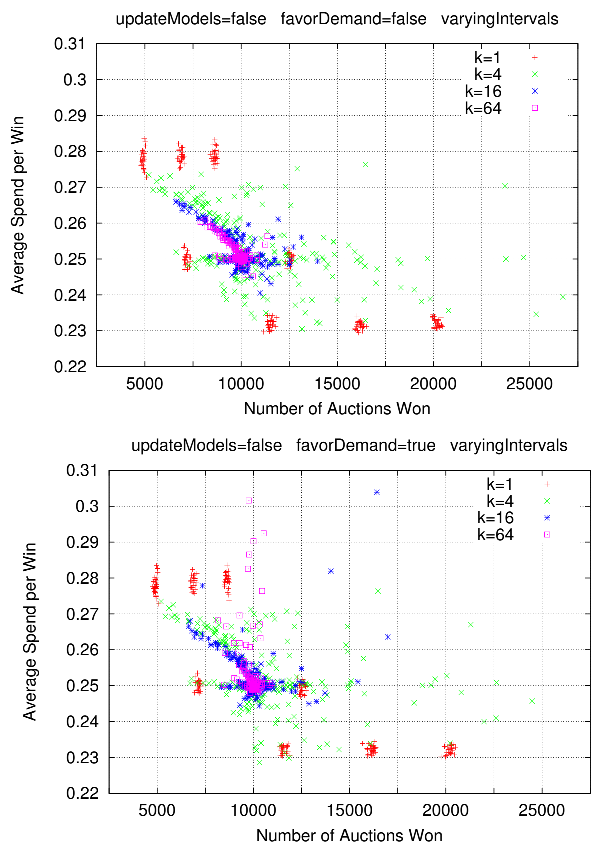


Figure 6: These plots are discussed in §7.2.

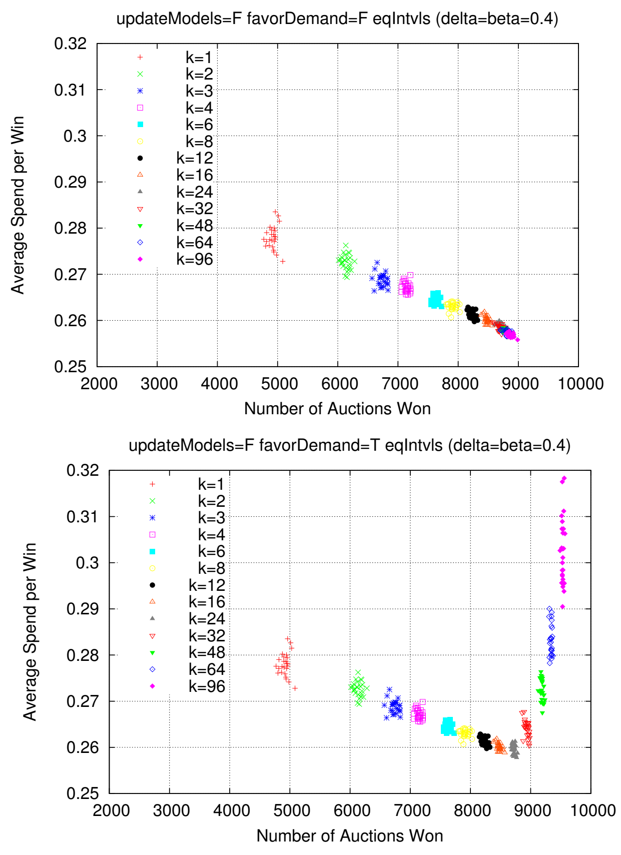


Figure 7: These plots are discussed in §7.2.

7.2 Investigation of Z > Y Case

In this section we investigate Question 4: what should the algorithm do when the demand-based bid z exceeds the spending-based bid y? This case can occur in reality, but it violates a simplifying assumption that was made during the theoretical analysis. The experimental investigation of this case involves new algorithmic details:

7.2.1 New Algorithmic Details for Z > Y Case

The *favorDemand* flag in the pseudocode of Figure 5 determines which bid is used in the assumption-violating "infeasible" case where the demand-based bid z is greater than the spending-based bid y. If favorDemand=true, then the algorithm will use the demand-based bid z, and will attempt to fulfill the demand even though that might result in overspending. If favorDemand=false, then the algorithm will use the spending-based bid y, and will avoid overspending but will tend to fall short of satisfying the demand. It is worth pointing out that in the "feasible" case $z \leq y$ (which was assumed in the theoretical analysis) there is no corresponding policy question because it is possible to simultaneously hit the demand and spending targets.

7.2.2 Experimental Results for Z > Y Case

To study the policy question for the $z > y$ case, we ran all of the experiments on synthetic data twice, once with favorDemand=false and once with favorDemand=true.

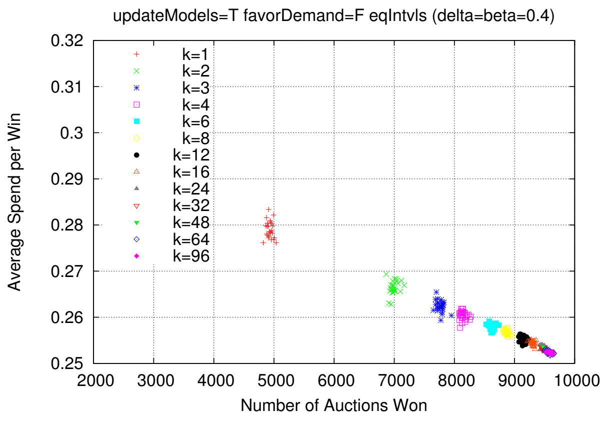
Some of the results are shown in Figure 6. Both panes contain scatter plots showing the outcomes of 25 runs each of the 9 error scenarios, with 4 different values of k, all with with unequal division of supply between time blocks. The top pane shows results for favorDemand=false, while the bottom pane shows results for favorDemand=true. The results are qualitatively similar, except for the larger number of magenta squares near the top of the bottom plot, indicating that some of the 25*9 runs for $k = 64$ ended up paying an excessive average price per auction win.

Further investigation showed that the runs in which the $k = 64$ runs overpaid were mostly for the $(\delta = 0.4, \beta = 0.4)$ error scenario, which for k=1 causes a bidder to underdeliver and over-pay per auction win.

To more clearly illustrate what is going on, consider the plots in Figure 7, which only contains results for the $(\delta = 0.4, \beta = 0.4)$ scenario, and which were obtained under the less-noisy simulation conditions where supply is equally divided between time intervals.

The top pane in Figure 7, which is for favorDemand=false, shows that as k is increased, the algorithm's demand achieved and average price paid initially moves towards the target values, but then seems to asymptote at values that fall short of the targets.

The bottom pane in Figure 7, which is for favorDemand=true, shows a very different behavior. The amount of demand satisfied continues to increase with increasing k, but the price paid stops decreasing and starts increasing around $k = 24$.



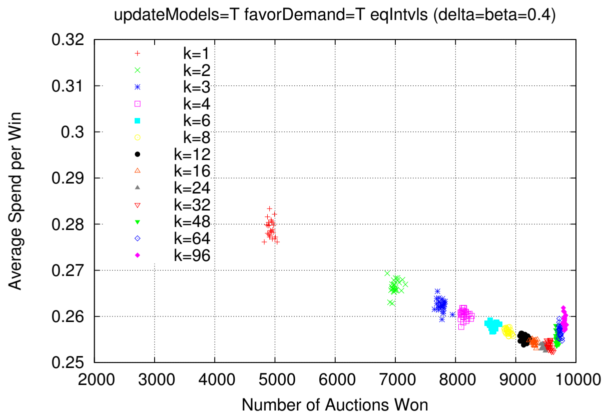


Figure 8: These plots are discussed in §7.3.

This is actually a general phenomenon that can occur whenever the following three conditions are met:

1. Large errors in the forecasts cause the algorithm to fall behind in satisfying the demand target.
2. Only a small fraction of the campaign's lifetime remains. This can only happen for larger values of k.
3. The algorithm is allowed to climb arbitrarily far up the price curve in pursuit of the faster win rates that would allow it to catch up.

Notice that the favorDemand=true policy can climb too far up the price curve, but the favorDemand=false policy cannot.

In summary, when forecast errors are such that the $z > y$ case arises, one can adopt the favorDemand=false policy and end up with some demand that is unsatisfied even for large k, or one can adopt the favorDemand=true policy and end up with an excessive average price that is exacerbated by large k.

It is in this situation, where increasing k does not strictly improve the outcome, that the idea of updating the erroneous forecasts begins to sound attractive. That idea is explored in the next section.

7.3 Updating the Forecasts

In this section we investigate Question 5: Can updating the forecasts yield better results? This involves some new algorithmic details.

7.3.1 *New Algorithmic Details for Updates*

The *updateModels* flag in the pseudocode of Figure 5 determines whether the supply and price models are updated after each time block. We note that the basic algorithm, as considered in all previous sections, corresponds to updateModels=false. This paper does not address the question of how to *optimally* update the models given the hypothetical feedback that would enable those updates. Instead, the models are updated in a very simple way that suffices to give some preliminary insight into question 5.

7.3.2 *Updating the Supply Model*

When updateModels = false, remaining supply is estimated as follows: (the variables here are explained in Table 9)

$$N_e = k_r \cdot \frac{N_f}{k}$$

When updateModels = true, remaining supply is estimated as follows:

$$N_e = k_r \cdot \frac{wN_f + N_a}{wk + k_a}$$

The rate at which observations overcome the forecast is affected by the parameter w. In these experiments, $w = 1$, so at the end of the simulation, the initial forecast and the observed data have approximately equal weight.

7.3.3 *Updating the Price Model*

The initial price model is represented by a set of samples drawn from the true price distribution and multiplied by the error factor $(1 - \beta)$.
If *updateModels* = false, this initial model is never changed, but if *updateModels* = true, the model is updated after each time block by unioning the model's current set of samples with the set of competing bids observed during that time block.
The rate at which observations overcome the forecast is affected by the size of the initial set of samples. In these experiments, there are 100000 samples in the initial forecast, and 100000 opportunities during the campaign, so at the end of the simulation the initial forecast and the observed data have equal weight.

7.3.4 *Experimental Results for Forecast Updates*

To study the question of whether a simple scheme for updating forecasts can yield improved performance, we ran all of the experiments on synthetic data twice, once with updateModels=false, and once with updateModels = true. In all, we performed 23400 simulations, which are (25 tries) * (9 error scenarios) * (13 values of k) * (2 supply subdivision schemes) * (2 values for favorDemand) * (2 values for updateModels). The general impression one obtains from looking at all of the resulting plots is that updating the models tends to give a tighter concentration of results around the target values, but not greatly so.

Due to space limitations, here we will just exhibit the plots in Figure 8, which correspond directly to the plots in Figure 7, except that the models are being updated in Figure 8.

A comparison of the top panes in the two figures, which are both for favorDemand=false, shows that the updating algorithm was able to get past the demand value of about 9000 at which the non-updating algorithm was asymptoting.

variable	meaning
D	Demand Target
D_r	Demand Remaining Unsatisfied
D_a	Demand Already Satisfied
B	Budget Target
B_r	Budget Remaining Unspent
B_a	Budget Already Spent
k	Number of Time Blocks
k_r	Time Blocks Remaining
k_a	Time Blocks Already Occurred
N_f	Forecasted Total Supply
N_e	Estimated Remaining Supply
N_a	Supply actually observed so far
$\mathcal{P}_f$	Forecasted Price Distribution
$\mathcal{P}_e$	Estimated Price Distribution
The following are hidden from the bidder.	
N^*	True total Supply
$\mathcal{P}^*$	True Price Distribution
δ	Supply Error: $N_f = N^* \cdot (1 + \delta)$
β	Price Error: $P_f = P^* \cdot (1 - \beta)$

Figure 9: Explanation of variables appearing in the pseudocode of Figure 5.

A comparison of the bottom panes in the two figures, which are both for favorDemand=true, shows that the average price paid by both algorithms starts to increase above a certain value of k. However, the average price paid for any given value of k is less for the updating algorithm.

8. CONCLUSION

Most online advertising today is sold via real-time auctions in which advertisers are represented by automated bidding agents that bid strategically in an attempt to achieve various goals including hitting demand targets and spending targets over a specified period of time. The inputs to these agents typically include machine-learned models of the world which facilitate extrapolation from the past to the future and from common events to rare events. Given their extrapolative function, it is not surprising that these models are imperfect.

While improving the accuracy of the models is a worthwhile research goal, in this paper, we have instead assumed that the models of future supply and future prices are inaccurate, and investigated the question of how a bidding agent can win the right number of impressions and spend the right amount of money given this inaccuracy.

In Theorem 4.1, we proved that, subject to certain feasibility conditions, a very simple bidding strategy, which only requires occasional feedback of the number of auctions actually won and the amount of money actually spent, quickly converges to the bidder's desired budget and demand.

We have also provided experimental evidence in Sections 5 through 7 that the proposed bidding strategy is robust enough to work even when some of the simplifying assumptions in the formal analysis are violated.

9. REFERENCES

[1] T. Chakraborty, E. Even-Dar, S. Guha, Y. Mansour, and S. Muthukrishnan. Selective call out and real time bidding. In A. Saberi, editor, *WINE*, volume 6484 of *Lecture Notes in Computer Science*, pages 145–157. Springer, 2010.

[2] P. Chen, W. Ma, S. Manalapu, C. Nagarajan, S. Vassilvitskii, E. Vee, M. Yu, and J. Zien. Ad serving using a compact allocation plan. submitted to WWW, 2012.

[3] Y. Chen, P. Berkhin, B. Anderson, and N. R. Devanur. Real-time bidding algorithms for performance-based display ad allocation. In C. Apté, J. Ghosh, and P. Smyth, editors, *KDD*, pages 1307–1315. ACM, 2011.

[4] Y. Cui, R. Zhang, W. Li, and J. Mao. Bid landscape forecasting in online ad exchange marketplace. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '11, pages 265–273, New York, NY, USA, 2011. ACM.

[5] N. R. Devenur and T. P. Hayes. The adwords problem: online keyword matching with budgeted bidders under random permutations. In J. Chuang, L. Fortnow, and P. Pu, editors, *ACM Conference on Electronic Commerce*, pages 71–78. ACM, 2009.

[6] J. Feldman, M. Henzinger, N. Korula, V. S. Mirrokni, and C. Stein. Online stochastic packing applied to display ad allocation. In M. de Berg and U. Meyer, editors, *ESA (1)*, volume 6346 of *Lecture Notes in Computer Science*, pages 182–194. Springer, 2010.

[7] J. Feldman, A. Mehta, V. S. Mirrokni, and S. Muthukrishnan. Online stochastic matching: Beating 1-1/e. In *FOCS*, pages 117–126. IEEE Computer Society, 2009.

[8] A. Ghosh, P. McAfee, K. Papineni, and S. Vassilvitskii. Bidding for representative allocations for display advertising. In Leonardi [12], pages 208–219.

[9] A. Ghosh, B. I. P. Rubinstein, S. Vassilvitskii, and M. Zinkevich. Adaptive bidding for display advertising. In J. Quemada, G. León, Y. S. Maarek, and W. Nejdl, editors, *WWW*, pages 251–260. ACM, 2009.

[10] R. M. Karp, U. V. Vazirani, and V. V. Vazirani. An optimal algorithm for on-line bipartite matching. In *STOC '90: Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 352–358, New York, NY, USA, 1990. ACM.

[11] K. Lang, J. Delgado, D. Jiang, B. Ghosh, S. Das, A. Gajewar, S. Jagadish, A. Seshan, C. Botev, M. Bindeberger-Ortega, S. Nagaraj, and R. Stata. Efficient online ad serving in a display advertising exchange. In I. King, W. Nejdl, and H. Li, editors, *WSDM*, pages 307–316. ACM, 2011.

[12] S. Leonardi, editor. *Internet and Network Economics, 5th International Workshop, WINE 2009, Rome, Italy, December 14-18, 2009. Proceedings*, volume 5929 of *Lecture Notes in Computer Science*. Springer, 2009.

[13] S. Muthukrishnan. Ad exchanges: Research issues. In Leonardi [12], pages 1–12.

[14] E. Vee, S. Vassilvitskii, and J. Shanmugasundaram. Optimal online assignment with forecasts. In D. C. Parkes, C. Dellarocas, and M. Tennenholtz, editors, *ACM Conference on Electronic Commerce*, pages 109–118. ACM, 2010.