2016 IEEE International Conference on Data Science and Advanced Analytics

Reserve Price Optimization at Scale

Daniel Austin
AppNexus
711 SW Alder St, Suite 400
Portland, Oregon 97205
Email: daustin@appnexus.com

Sam Seljan
AppNexus
711 SW Alder St, Suite 400
Portland, Oregon 97205
Email: sseljan@appnexus.com

Julius Monello
AppNexus
711 SW Alder St, Suite 400
Portland, Oregon 97205
Email: jmonello@appnexus.com

Stephanie Tzeng
AppNexus
28 W 23rd St.
New York, New York 10010
Email: stzeng@appnexus.com

***Abstract*—Online advertising is a multi-billion dollar industry largely responsible for keeping most online content free and content creators ("publishers") in business. In one aspect of advertising sales, impressions are auctioned off in second price auctions on an auction-by-auction basis through what is known as real-time bidding (RTB). An important mechanism through which publishers can influence how much revenue they earn is reserve pricing in RTB auctions. The optimal reserve price problem is well studied in both applied and academic literatures. However, few solutions are suited to RTB, where billions of auctions for ad space on millions of different sites and Internet users are conducted each day among bidders with heterogenous valuations. In particular, existing solutions are not robust to violations of assumptions common in auction theory and do not scale to processing terabytes of data each hour, a high dimensional feature space, and a fast changing demand landscape. In this paper, we describe a scalable, online, real-time, incrementally updated reserve price optimizer for RTB that is currently implemented as part of the AppNexus Publisher Suite. Our solution applies an online learning approach, maximizing a custom cost function suited to reserve price optimization. We demonstrate the scalability and feasibility with the results from the reserve price optimizer deployed in a production environment. In the production deployed optimizer, the average revenue lift was 34.4% with 95% confidence intervals (33.2%, 35.6%) from more than 8 billion auctions over 46 days, a substantial increase over non-optimized and often manually set rule based reserve prices.**

***Keywords*-Reserve Price; Revenue Optimization; Online Advertising; Real-Time Bidding**

I. Introduction

Online advertising is a multi-billion dollar industry with $27.5 billion in revenue generated in the first half of 2015 alone [1]. Online advertising space is sold in many ways, such as through direct sales of a guaranteed number of impressions [2], [3], on ad networks or with header-bidding [4], or via real-time bidding (RTB) at auction on ad exchanges [5], [6], [7]. Publisher monetization itself is a big business with several companies known as supply side platforms (SSPs) offering services to publishers to increase monetization. Monetization can be improved in a number of different ways; in this paper, we focus on setting optimal reserve prices in RTB auctions. In RTB, most auctions are second-price with reserve [8]. This means that the highest bidder wins only if they bid higher than the reserve price, and the price the winner pays is the greater of the second highest bid price and the reserve price. Care must be taken in setting the reserve price because if set too low, the reserve will have no effect on auction revenue and if set too high, the auction will not have a winner and the publisher will not gain revenue. Thus, reserve price optimization is one of the main mechanisms through which publishers can directly influence revenue without substantively changing the user experience [9]. The reserve price problem is especially difficult in large ad-exchanges because billions of auctions are transacted each day. A good reserve price optimizer needs to be simple enough that it can compute a reserve price in a few milliseconds at auction time, but also complicated enough to take advantage of the most recently available auction-level data.

To the best of our knowledge, we present the first online, scalable, real-time reserve price optimizer that: 1) dynamically sets reserve prices and learns it's parameters on an auction-by-auction basis, and 2) provably scales to billions of auctions per day in a production environment. Our solution applies a learning approach, maximizing a custom cost (revenue) function suited to reserve price optimization. Our cost function is not convex, so we employ a global search strategy to augment stochastic gradient decent. We show that this reserve price optimizer improves revenue from RTB auctions by an average of 34.4% with 95% confidence interval of (33.2%, 35.6%) over an average of 181.4 ± 46.6 million auctions per day for a total of over 8 billion analyzed auctions in a 46 day period. Moreover, the production implementation of the algorithm used a total of 10 servers (two for each of five regional data centers), underscoring the resource cost scalability of our approach.

The rest of the paper is structured as follows: in section 2 we survey other approaches toward reserve price optimization. In section 3 we formally state the reserve price optimization problem and outline the solution. In section 4 we present results demonstrating the efficacy of the proposed methodology. This is followed in section 5 with a discussion of the advantages and disadvantages to the proposed approach along with suggested future work. In section 6 we conclude with a recap of the contributions of this paper.

978-1-5090-5206-6/16 $31.00 © 2016 IEEE
DOI 10.1109/DSAA.2016.32

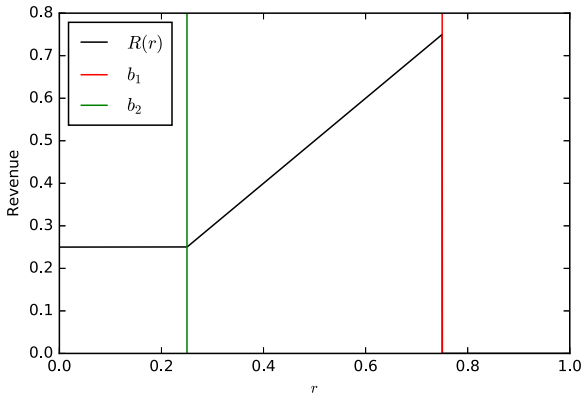

Figure 1. Single auction revenue, R, as a function of the reserve, price, r, with first and second price fixed at $b_2 = \$0.25$ and $b_1 = \$0.75$.

II. Related Work

Reserve price optimization is a well known problem that has been thoroughly investigated in the academic literature. A second price auction with reserve has been shown to be optimal with a single object for sale and an independence assumption on bidder valuations [10]. This auction mechanism has an analytical solution for the reserve price, r, in the simple case of independent and identically distributed bids:

$$r = \frac{1 - F(r)}{f(r)} + v_s, \tag{1}$$

where F and f are the cumulative and probability distributions of bidder valuations of the object being auctioned, respectively, and v_s is the value of the object to the seller if it goes unsold. One additional condition is needed, and that is that the hazard rate $h(r) = \frac{f(r)}{(1-F(r))}$ must be strictly increasing. These somewhat restrictive assumptions have been relaxed and the results have been extended in a number of ways, for example in [11], [12].

Unfortunately, this approach is problematic in the context of RTB auctions. Although we can estimate bidder's private valuations from empirical bid distributions, the actual distributions violate the core assumptions required to use this equation: they are not independent and identically distributed across bidders. When valuations are not identically distributed, the optimal reserve mechanism calls for different reserve price estimates per bidder [8]. However, setting different reserve prices per bid was not possible for us to implement for a number of reasons including product definition and technical implementation challenges. If reserve prices are not stratified by bidder, then bid distributions are best approximated by some mixture distribution, which often violates the assumption that the hazard rate is increasing.

Moreover, learning how the market of advertisers are likely to bid is the most important element of the reserve price optimization problem in RTB. RTB auctions vary by website, the location of the ad placement on the page, ad placement size, media type (banner or video), hour of the week, season, the user who will view the ad, and how many ads the user has seen (known as session depth). On the AppNexus Platform millions of ad placements and hundreds of millions of unique users are seen each day. Even for the same user on the same site, bids vary greatly over time as advertising campaigns start and end, and users' web surfing behavior evolves. A good discussion of these and other problems within ad exchanges are in [13] and [14].

As a result, there has been increasing interest in studying the reserve price problem for online auctions, and specifically for RTB. A moderately sized study of almost 500,000 auctions showed that applying optimal auction theory principles to set the reserve in a sponsored search setting resulted in a small estimated increase in revenue of close to 3% and as much as 10% for certain keywords compared to a heuristic reference on real data [15]. The approach did not appear to modify reserve prices on an auction-by-auction basis as would be ideal for RTB, but provided an approach that scaled to the large number of auctions conducted daily. Another recent study compared a number of different mechanisms to set the reserve price - some based on auction theory, some based on private value, and the so-called "one-shot" algorithm [14]. The one-shot algorithm adaptively adjusts the reserve up slightly if it was too low in prior auctions and adjusts it down substantially if it ever was set too high, similar to how the "bold driver" algorithm works in online learning [16]. The authors report normalized results indicating that the one-shot algorithm performed better in most cases than the other implemented algorithms in a mix of offline and limited online testing. While promising, it's not clear how to generalize the approach to best take advantage of a high dimensional feature space. Another recent algorithm using regret minimization has an attractive bound on the regret of $\widetilde{O}(\sqrt{T})$ (T is the number of auctions) using only prior auction outcomes [17] but makes the somewhat limiting assumption that all bidders draw from the same unknown distribution, which is an untenable assumption for online auctions on ad exchanges.

More recently, some approaches have treated reserve price optimization as a learning problem, utilizing auction features in addition to prior auction bids or revenues to set reserve prices. This general approach seems the most promising as it permits the use of all available information to optimally set a reserve while also providing a framework to avoid overfitting historical data. In the first example, the reserve price problem is cast as an optimization problem that can be solved with Difference of Convex functions (DC) programming, and algorithms are provided to solve the optimization problem [9]. The results indicated solid performance on simulated data, and the results were extended and applied to a small data set of about 70,000 eBay auctions [18]. The results here are

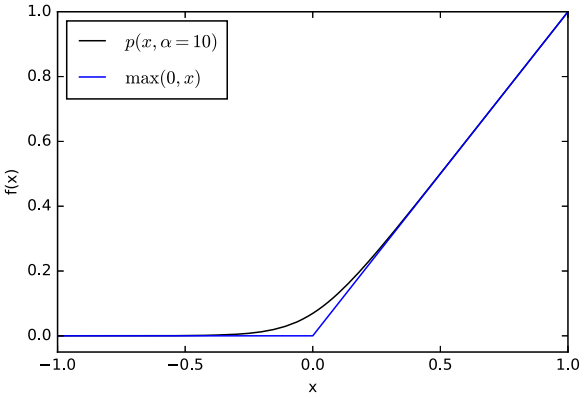

Figure 2. Continuous approximation to $\max(0, x)$ by the so called plus function, $p(x, \alpha)$.

impressive, but the algorithm is computationally expensive - especially at our scale - and would need to be modified to address overfitting. The approach most similar to our own uses a smoothed version of the non-convex revenue function (defined formally below) to approximate the cost function used to estimate reserve prices [19] using either a linear or non-linear functional form for the reserve estimator. This method was validated on a combination of synthetic data and the same small eBay data set of 70,000 observations in [9]. They report strong performance on this data, but it is not clear how their algorithm would scale to RTB auctions with its massive data flow, dynamic auction environment and time-critical performance requirements. In general, with the exception of [15] and possibly [14] it appears that most published approaches are more suitable to eBay style online auctions occurring less frequently and on slower time scales than the billions of daily RTB auctions each occurring in less than 100 ms each on ad exchanges.

III. Optimal Reserve Price

We first describe the problem formulation and solution generically, followed by a description of specific implementation details.

A. Problem Formulation

A reserve [20] is often defined as the minimum amount a seller is willing to accept for an item, but in the context of revenue can also be thought of as a proxy bid. That is, revenue, R, with a reserve r, for a single auction with first (highest) and second (highest) bids b_1 and b_2, respectively, is described by:

$$R(b_1, b_2, r) = \begin{cases} b_2, & \text{for } r \leq b_2 \\ r, & \text{for } b_2 < r \leq b_1 \\ 0, & \text{for } b_1 < r \end{cases} \tag{2}$$

The revenue function is plotted for illustrative purposes in Fig. 1 for fixed values of b_1 and b_2 while varying r. What Eq. 2 and Fig. 1 suggest is that if a reserve price is set between the highest and second highest bids, price reduction will diminish and publisher revenue will increase.

On many ad exchanges such as AppNexus, billions of auctions are transacted each day. The problem is therefore to find a sequence of reserve prices that maximizes the second-price auction revenues with reserve for a given publisher:

$$\mathbf{r} = \text{argmax}_{\{r_i\}} \sum_i R(b_{i1}, b_{i2}, r_i) \tag{3}$$

where index i ranges over all auctions. If the reserve prices did not affect bids and we knew ahead of time what the bids were, the problem would be trivial because we could use the highest bid as the reserve price for each auction and maximize overall revenue. However, the bids are not available ahead of time; thus, we must estimate the reserve price based on some prediction of the bids. Additionally, the auctions occur sequentially suggesting that an approach with online, auction-by-auction incremental updates would take advantage of new data as soon as it is available to inform the algorithm.

B. Problem Solution

Similarly to [18] and [19], we formulate the reserve price estimation problem as a learning problem with the goal of estimating the first price of each auction on an auction-by-auction basis. After some research and consultation with engineering we modeled [21] the reserve price as a linear function of auction attributes and to use stochastic gradient ascent [22] to update the model parameters after each auction. More formally, we estimate the reserve price for an arbitrary auction as:

$$\hat{r} = \mathbf{x}^T \boldsymbol{\beta} \tag{4}$$

where $\mathbf{x}$ is a vector containing auction descriptors from before the bids have been placed, and β is a vector of the model parameters. The model parameters can be updated sequentially by stochastic gradient ascent as:

$$\boldsymbol{\beta}_{i+1} = \boldsymbol{\beta}_i + \eta \nabla R(\boldsymbol{\beta}_i) \tag{5}$$

where i indexes the sequentially occurring auctions, η is a suitably chosen learning rate discussed in more detail below, and $\nabla R(\boldsymbol{\beta}_i)$ is the derivative of the revenue function with respect to the current $\boldsymbol{\beta}$. However, as shown in Eq. 2, the revenue function is not differentiable, and Eq. 5 cannot be directly applied. To overcome this we used some results from the theory of convex approximation to remove the discontinuities from the revenue function. There are two discontinuities in Eq. 2 (see also Fig. 1) that need to be "smoothed." The first occurs at the transition in revenue when the reserve price first exceeds the second price. This can be approximated by what is commonly referred to as

the "plus" function [23] as shown in Fig. 2. More formally, we have:

$$p(x, \alpha) = x + \frac{1}{\alpha}\log(1 + e^{-\alpha x}) \approx \max(0, x) \tag{6}$$

with α a smoothing parameter that controls the smoothness of the approximation. The second discontinuity occurs at the transition when the reserve exceeds the first price and is a step discontinuity. This can be approximated with a (reversed) sigmoid function also parameterized by α. This is shown in Fig. 3 and is represented as:

$$s(x, \alpha) = 1 - \frac{1}{1 + e^{-\alpha x}} \tag{7}$$

Combining these two approximations into a single function and shifting by the first and second highest bids yields the following somewhat unwieldy but useful approximate revenue function (also represented graphically in Fig. 4):

$$\begin{aligned}\hat{R}(b_1, b_2, r, \alpha) =& r + \frac{1}{\alpha}\log(1 + e^{-\alpha(r-b_2)}) \\ &- (r - b_1 + \frac{1}{\alpha}\log(1 + e^{-\alpha(r-b_1)})) \\ &- \frac{b_1}{1 + e^{-\alpha(r-b_1)}}\end{aligned} \tag{8}$$

Taking the derivative of this function with respect to β give the final piece needed for online gradient ascent:

$$\begin{aligned}\nabla\hat{R}(\boldsymbol{\beta}) =& \mathbf{x}\Bigg(- \frac{\alpha b_1 e^{-\alpha(r-b_1)}}{(1 + e^{-\alpha(r-b_1)})^2} + \frac{e^{-\alpha(r-b_1)}}{1 + e^{-\alpha(r-b_1)}} \\ &+ \frac{e^{-\alpha(r-b_2)}}{1 + e^{-\alpha(r-b_2)}}\Bigg)\end{aligned} \tag{9}$$

where we have omitted the explicit dependence of $\nabla\hat{R}$ on b_1, b_2, r, and α for notational simplicity. Putting this all together gives the complete stochastic gradient ascent update as:

$$\boldsymbol{\beta}_{i+1} = \boldsymbol{\beta}_i + \eta\nabla\hat{R}(\boldsymbol{\beta}_i) \tag{10}$$

C. Practical Considerations

The reserve price estimator model learned with prediction and correction updates is described by Eqs. 4 and 10, respectively. In our implementation, we learn a model for each publisher who chooses to use reserve optimization. This allows data collected from a publisher's own auctions to inform optimal reserve pricing. In addition to implementing a model for each publisher, there are many other practical considerations when implementing the proposed reserve optimizer in a production system. In this section we discuss several of them.

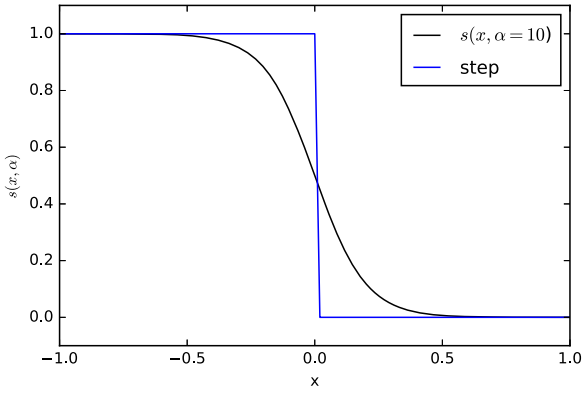

Figure 3. Continuous approximation to a reversed step discontinuity by a sigmoid, $s(x, \alpha)$, for $\alpha = 10$.

1) Feature Engineering and Selection: A core advantage of our algorithm is the fact that it can use any type and number of features. At the same time, algorithm performance is heavily dependent on the features that are used. There are many raw auction-based features available to publishers on the AppNexus platform. In the implementation we describe here, we use a total of 10 features. Since the calculations are specific to the AppNexus data pipeline, we describe the features qualitatively instead of quantitatively. The features are of three types: 1) attributes of the online content (e.g. web site, ad placement, ad size), 2) the type of user viewing the ad - referred to as a segment, and 3) the behavior of the user. We did not use any features that are specific to buyers. Although features describing attributes of the buyers are valuable from a predictive standpoint, they require returning multiple reserves per auction, which was not technically possible when we implemented the reserve price optimizer.

We distinguish between 3 types of internet users who appear on our platform - those with cookies (we have seen them before), those that we have not seen before (cookieless, meaning a cookie is allowed but has not been set), and those that do not allow cookies (thus we are uncertain if we have seen them before or not). For users who have been seen on the platform before, we used six features. Two of the features estimate each user's segment value - a recent history of their segment's value and a longer-term average. Two of the features describe the frequency of a user being seen by AppNexus buyers on the publisher's website. The final two features are a constant vector and a dynamic estimate of the valuation of the publisher's inventory, which varies at the placement and country level. For users who are new and those who are cookieless, the features are a constant vector and the dynamic inventory valuation. 10 features is a relatively small number, but using segment and inventory aggregates allowed us to capture bidding

information efficiently.

During the feature selection phase, we tried several other features such as placement size, country of the user, and many others. These features were all excluded as non-informative when including the described features. We also explored using time-related features (hour of day, day of week, etc.) but found that our time-varying user valuations and frequency measures captured most of the relevant fluctuations that might have been measured by more explicit time-related features.

2) Data Preprocessing: Data should be preprocessed to be scaled and possibly centered. We found little difference in algorithmic performance between a multiplicative variable scaling approach (e.g., scaling variables to be approximately in the range from 0 to 1) and the more standard centering and scaling approach [24]. We chose the multiplicative approach because it required storing fewer values and less computation in production, although the savings were fairly minimal. The scaling values were chosen using an optimization procedure described in more detail below.

3) Model Initialization: In an online production implementation, the model can and should be initialized with $\boldsymbol{\beta}$ values that are reasonable. This allows the reserve price optimizer to set decent reserve prices when little data have been seen, and improves the speed of convergence to the true optimum. In our implementation, we optimized the starting $\boldsymbol{\beta}$ values over several different publishers using an offline analysis to identify values that would perform well for all publishers. We then apply these same initial values to any new publisher models when deployed.

4) Regularization: While the update rule in Eq. 10 forms the the foundation of our solution, another important aspect of implementation is to prevent overfitting of the model parameters. The model parameters that provide the greatest revenue in training data do not necessarily provide the greatest revenue in production because historical bids are an imperfect estimate of future behavior. This is somewhat mitigated by using stochastic gradient ascent, but we generalized the solution by adding regularization to the gradient update (Eq. 10). In particular, we use $||L||^2$ regularization [25] in the gradient update:

$$\boldsymbol{\beta}_{i+1} = \boldsymbol{\beta}_i + \eta(\nabla \hat{R}(\boldsymbol{\beta}_i) - 2\boldsymbol{\mu} \odot \boldsymbol{\beta}_i) \tag{11}$$

where $\odot$ represents element-wise multiplication, and $\boldsymbol{\mu}$ is a vector of regularization parameters determined by an optimization procedure described in more detail below.

5) Learning Rate: Setting the learning rate η correctly is important – too large a value and the model will never converge when auction dynamics are stationary, but too small of a value and changing dynamics cannot be quickly or adequately tracked. In the first implementation of our approach, we found that a value of $\eta = 0.0001$ worked well over several publishers and days of data. However, a static learning rate is somewhat restrictive and much work has been done to adaptively tune learning rates, such as search then converge strategies [26], momentum based approaches [27], and ADAGRAD [28]. None of these methods seem to adapt particularly well for the case of stationary dynamics, non-stationary dynamics, or for a non-convex cost function.

As a result, we developed a custom procedure for selecting different learning rates over time that reliably improved performance over using a static learning rate. In particular, we employed the following procedure: 1) Run the optimizer with M different learning rates in parallel, 2) Record the cumulative auction revenue, $R(\eta_k)$, for each of the M learning rates over a series of auctions where $k = 1, ..., M$, 3) For every N auctions, check and see which learning rate generated the most revenue in the last N auctions, and 4) If the current learning rate is not the best, switch to the best learning rate (select η_k, where $k = \text{argmax}_k R(\eta_k)$). A block diagram of this procedure in the context of the whole implementation is shown in Fig. 5. We selected $M = 10$ as a tradeoff between storage, computation, and model learning performance. We selected N to be 12,500 using an optimization procedure described below.

6) Hyper-parameters: There are 23 hyper-parameters in our model including the learning rate, the smoothness of the revenue approximation (α), the regularization parameters, and feature scaling parameters. In the results we present below, we used the same set of parameters for all publishers, which we view as a strength of our approach.

These hyper-parameters must be carefully selected in order to work well over time and for different publishers. To select the hyper-parameters, we implemented a 3-tiered procedure. First we jointly optimized the 23 parameters over data collected from 12 publishers over a two day period. The cost function for the optimization was the total sum of the revenue across publishers, where the revenue for each publisher was calculated by sequentially processing the data with the reserve optimizer. Because the revenue function is non-convex (it is the sum of many unimodal revenue functions), the optimization was deployed in two parts: particle swarm optimization [29] was utilized to reduce the search space to a handful of "good" starting points, and Powell's method [30] was used to then find the associated local optimum. The best point (23-vector) of parameters overall was taken as the optimum.

After finding the optimum, we tested the reserve optimizer on an out-of-sample data set collected 2 months later from another 12 publishers. This data was processed with the optimal meta-parameters and compared to default parameters that were selected as working values during preliminary algorithm development. The improvement in revenue seen was 4.85% on the validation set. Finally, the optimal parameters were pushed to production and validated across the platform where the approximately 5% improvement in revenue was shown to hold. As a final recommendation,

we suggest implementing metrics to track revenue and to periodically refit/validate the meta-parameter estimates. This ensures that platform level changes are detected and the parameters can be updated if needed.

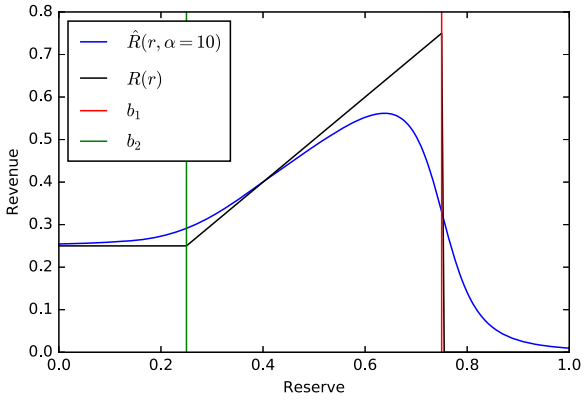

Figure 4. Approximate ($\hat{R}(r, \alpha)$) and actual ($R(r)$) revenue function for $\alpha = 10$, with first and second price fixed at $b_2 = \$0.25$ and $b_1 = \$0.75$.

IV. Experiment and Results

In this section, we present the results of two experiments. The first is on a large set of real auction data collected from AppNexus' bid landscape that were analyzed offline in Python (version 2.7) to simulate the production environment. The second are the results from 46 days of the reserve optimizer deployed in AppNexus' production environment. The offline analysis was conducted based on 2 days of bid landscape data collected from the AppNexus platform in November of 2015 from 12 publishers for a total of 25,152,667 transacted auctions. The bid landscape data consisted of the first and second bid prices, user-based auction features, and other data not used in this experiment. Code was developed and prototyped in Python to implement the reserve optimizer described above using older bid landscape data collected in July of 2015, thus the results here are an out-of-sample validation. That is, the algorithm was applied and incrementally updated as described above but no parameters were tuned on the November 2015 data.

The production results are not experimental, as they are actual results recorded from our reserve optimization product currently offered as part of the AppNexus Publisher Suite. The production implementation was written and deployed in Java. Data from production is presented from 46 days spanning 2/8/2016 - 3/24/2016. For both sets of results, where 95% confidence intervals are reported they are calculated with the bootstrapping technique [31] due to our hesitance to make distributional assumptions on the underlying data. A 95% confidence interval (CI) not containing 0 indicates statistical significance at the 5% level and is analogous to the two-sided bootstrap test which rejects the null hypothesis that the test statistic is 0.

For the offline experiment, the publisher data has been anonymized and relabeled with a publisher ID from 1-12. There was an average of 2,096,056 auctions per publisher with a range of 57,014 to 6,985,126. The mean revenue lift compared to a pure second price auction was 21.5% with a 95% CI of (8.3%, 37.6%). Note that this metric is an average lift per publisher. As can be seen in Table I, the lift varies across publishers with some publishers benefitting substantially and others not benefitting at all. Reasons for this are discussed in more detail in the Discussion section.

The average daily number of production auctions with optimal reserve applied was 181,428,936 with a range of 102,354,923 to 277,202,594 for a total of 8,345,731,064 reserve optimized auctions over 46 days. The average revenue lift compared to no optimal reserve applied was 34.4% with a range of (27.9% - 45.8%) and a 95% bootstrap confidence interval of (33.2%, 35.6%). Note that this metric measures total revenue lift across all publishers. These results are shown graphically in Fig. 6. Not every publisher chooses to use the described reserve optimizer and not all auctions are open RTB auctions, so the number of optimized auctions is smaller than the total number of auctions transacted on AppNexus' platform.

V. Discussion

The results indicate robust, statistically significant daily revenue lift of about 35% on average using the proposed reserve optimizer. The results were collected in a large-scale, online production deployment conducted over an extended time. The main advantages of our algorithm are threefold: 1) it is fast and scalable, 2) it can both set reserve prices on an auction-by-auction basis using individual auction-level data, 3) it can be incrementally updated between auctions for upwards of hundreds of millions of auctions per day. The stochastic nature of the updates both allows for convergence to near optimal reserve prices under stationary auction dynamics and allows the reserve estimator to track changing dynamics over time. To the best of our knowledge, this is the largest study reported to date with production results from over 8 billion reserve optimized auctions. It is also the only study with an online and incrementally updated algorithm deployed at scale in a production environment. In addition, our solution uses a minimal amount of computing resources, both in computation and storage. We process bid level auction data on 10 servers, writing only monitoring aggregates to disk. These results are very promising, but there are some issues that deserve discussion.

First, as seen in Table I, some publishers benefit more than others from reserve optimization (e.g., compare the lift between publishers 3 and 9). This is due to a number of factors such as lower bid density or heterogeneous valuation among the bidders. In both cases, there tends to be a greater

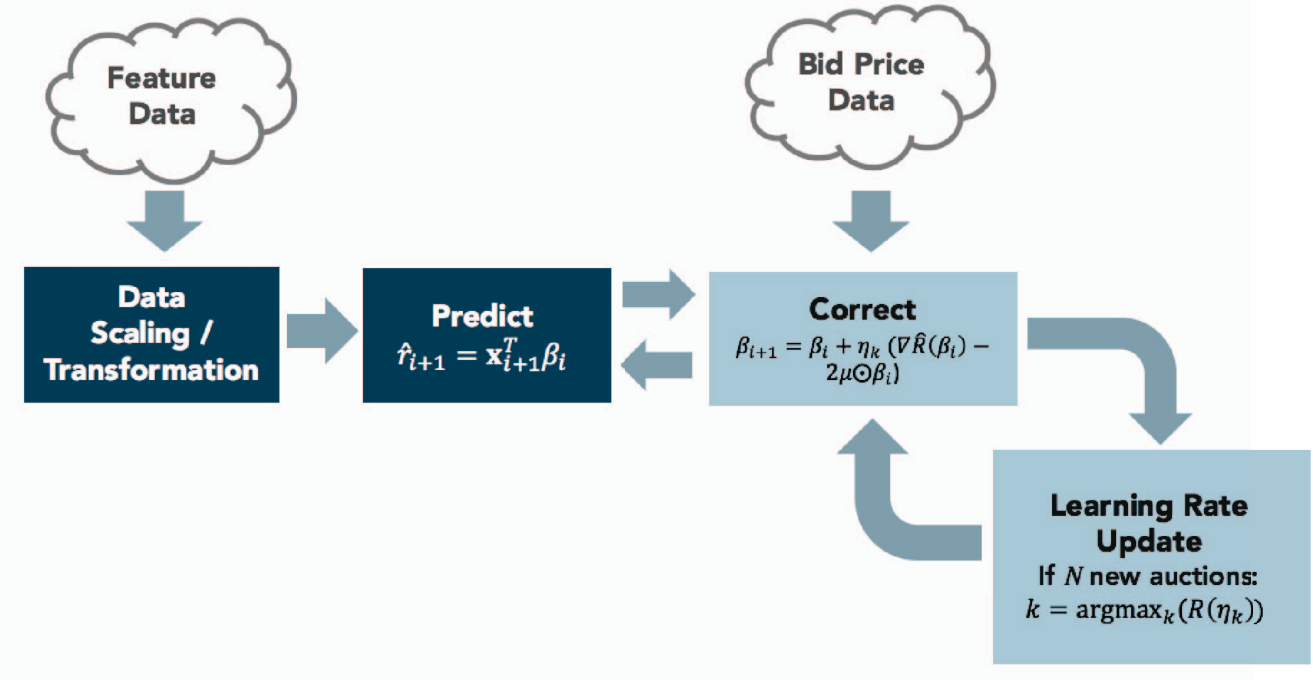

Figure 5. Block diagram of the complete reserve optimizer system.

Table I
NUMBER OF AUCTIONS AND REVENUE LIFT OVER A PURE SECOND PRICE AUCTION FOR OFFLINE AND OUT-OF-SAMPLE ANALYSIS ON DATA COLLECTED FROM 12 PUBLISHERS USING PYTHON SIMULATED ENVIRONMENT. CI IS CONFIDENCE INTERVAL.

Publisher	1	2	3	4	5	6	7	8	9	10	11	12	Mean (95% CI)
Number of Auctions	517,201	794,818	6,331,090	57,014	385,854	2,108,042	2,462,738	610,683	285,473	3,489,631	6,985,126	1,124,997	2,096,055 (962,624 - 3,561,572)
% Lift	5.07%	60.56%	78.96%	4.63%	1.03%	17.00%	43.43%	0.60%	-0.28%	9.89%	2.11%	13.8%	21.53% (7.96% - 37.58%)

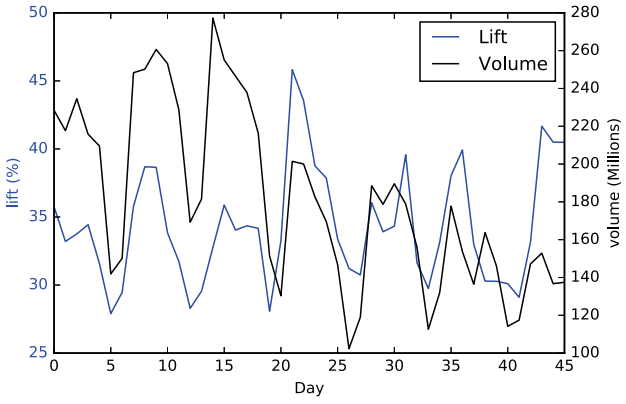

Figure 6. Revenue lift (blue trace) and impression volume in millions (black trace) for the described reserve optimization implemented in production. This figure spans a 46 day period from 2/8/2016 - 3/24/2016.

disparity between the first and second prices of any given auction, thus giving more opportunity for a reserve price to capture additional auction revenue. Therefore, publishers with lower bid density and larger differences in bidder valuations benefit substantially more from reserve optimization than others.

Another important consideration is in regard to the approximate revenue function described by Eq. 8 and shown in Fig. 4. As can be seen in Fig. 4, the maximum value of the true and approximate revenue functions do not coincide. This does cause some problems with accuracy in that the model will adapt towards a maximum that is lower than the true revenue maximum. The effect is a conservative reserve estimator. However, recall that the cost of overestimating the optimal reserve is loss of all revenue, while the cost of underestimating is a linear loss of revenue. This suggests that conservatism is not too large of a problem - we gladly trade some loss of potential revenue due to lower reserves to help ensure fewer overestimated reserve prices. Furthermore, the maximums can be made arbitrarily close to each other by increasing the smoothing parameter α, but doing this will make the derivative of the approximate revenue function too large near the original discontinuities. This in turn can cause a huge gradient step away from the desired maximum. Ultimately, the α that is chosen allows one to balance

between approximation and stability.

The choice of a linear model also poses some shortcomings to our approach. The most notable is that our proposed reserve optimizer can return negative values. A negative value does not make sense as a reserve price, but from a practical point of view is no more harmful to revenue than a reserve price set lower than the second price. In addition, all publishers impose a minimum reserve price, so we never apply a negative reserve to any auction. Some options to mitigate this include transforming the bid prices via the log or box-cox transform [32] to avoid the embarrassment of negative reserves. However, we have found that a linear model in log space performs worse than a linear model in regular space for this problem. Other nonlinear models could solve this problem, but we have yet to uncover an approach that outperforms the linear model and can be deployed at scale. The assumption of linearity on the model parameters is also almost certainly wrong, but we knowingly trade accuracy for the speedy calculation of reserve estimates.

The proposed reserve price optimizer seeks to maximize revenue. However, in some cases, publishers have goals other than pure revenue maximization. For example, a publisher may seek to maximize revenue while minimizing the number of ads displayed on screen. This may seem counterintuitive, but fewer ads tends to improve user experience, and long term publisher revenues are affected by user experience. One way to do this would be to increase the aggressiveness of the proposed reserve optimizer so that higher reserves are set in general. This tends to increase revenue per ad filled while decreasing the number of ads that were displayed. Augmenting this approach with a simple adaptive target would allow the aggressiveness to be explicitly controlled, but a complete description of such a modification is outside the scope of this paper.

Another potential concern is that setting reserve prices too close to the first price may result in a noticeable change in auction dynamics. That is, if bidders noticed a lack of price reduction in auctions in which they won, they may change behavior and start bid shading (bidding less than their valuation), which is a dominant strategy in first price auctions [33], [8]. Two different aspects of our approach help prevent this. First, as discussed above, the maximum of the approximate revenue function is lower than the true revenue function, so the optimal reserve does not change auctions from a second price to first price mechanism. The second aspect is that the collection of features used and model together are not accurate enough to perfectly predict bid prices. This can be seen by noting that different bidders have different valuations for the same ad space up for auction, but our model learns the bids of the market as a whole and returns a single reserve for each auction. As a result, there is little risk in changing auction dynamics with this proposed reserve optimizer.

VI. Conclusions

In this paper, we present what we believe to be the first example of a reserve price optimizer that has been provably deployed in an actual production environment. We demonstrated an average daily lift of 35% across billions of auctions in an online and real time production implementation. We described the algorithm, gave suggestions for how to implement it in a production environment, and reported results both offline (on a small group of publishers) and online (for all publishers who use reserve optimization on our ad exchange). We further discussed opportunities for improvement in the reserve optimizer. Future work will consist of improving the prediction of reserve prices by experimenting with non-linear models that may be approximately updated in real time. Also, we will further experiment with non-linear feature transformations to try and capture some of the non-linearities inherent in auction dynamics in our linear model. Finally, we will investigate which auction-level features are most predictive of optimal reserve pricing.

Acknowledgment

The authors would like to thank Mike McNeeley, Ron Lissack, Sandesh Devaraju, Mark Ha, Andrew Sweeney, Yoandy Terradas-Otero, Brett Carter, and John Paul Wallaway for helping design and implement the reserve optimizer in production and creating the monitoring tools to gather the results presented herein.

References

[1] PwC & IAB, "Iab internet advertising revenue report: 2015 first six months results," PwC & IAB, An industry survey conducted by PwC and sponsered by the Interactive Advertising Bureau (IAB), October 2015.

[2] B. Chen, S. Yuan, and J. Wang, "A dynamic pricing model for unifying programmatic guarantee and real-time bidding in display advertising," in *Proceedings of the Eighth International Workshop on Data Mining for Online Advertising*, ser. ADKDD'14. New York, NY, USA: ACM, 2014, pp. 1:1–1:9. [Online]. Available: http://doi.acm.org/10.1145/2648584.2648585

[3] V. Bharadwaj, P. Chen, W. Ma, C. Nagarajan, J. Tomlin, S. Vassilvitskii, E. Vee, and J. Yang, "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*, ser. KDD '12. New York, NY, USA: ACM, 2012, pp. 1195–1203. [Online]. Available: http://doi.acm.org/10.1145/2339530.2339718

[4] S. Sluis, "The rise of 'header bidding' and the end of the publisher waterfall," http://adexchanger.com/publishers/the-rise-of-header-bidding-and-the-end-of-the-publisher-waterfall/, adexchanger.com, June 2015. [Online]. Available: http://adexchanger.com/publishers/the-rise-of-header-bidding-and-the-end-of-the-publisher-waterfall/

[5] J. Wang and S. Yuan, "Real-time bidding: A new frontier of computational advertising research," in *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, ser. WSDM '15. New York, NY, USA: ACM, 2015, pp. 415–416. [Online]. Available: http://doi.acm.org/10.1145/2684822.2697041

[6] K.-C. Lee, A. Jalali, and A. Dasdan, "Real time bid optimization with smooth budget delivery in online advertising," in *Proceedings of the Seventh International Workshop on Data Mining for Online Advertising*, ser. ADKDD '13. New York, NY, USA: ACM, 2013, pp. 1:1–1:9. [Online]. Available: http://doi.acm.org/10.1145/2501040.2501979

[7] S. Yuan, J. Wang, and X. Zhao, "Real-time bidding for online advertising: Measurement and analysis," in *Proceedings of the Seventh International Workshop on Data Mining for Online Advertising*, ser. ADKDD '13. New York, NY, USA: ACM, 2013, pp. 3:1–3:8. [Online]. Available: http://doi.acm.org/10.1145/2501040.2501980

[8] V. Krishna, *Auction theory*. Academic press, 2009.

[9] M. Mohri and A. M. Medina, "Learning theory and algorithms for revenue optimization in second-price auctions with reserve," *arXiv preprint arXiv:1310.5665*, 2013.

[10] R. B. Myerson, "Optimal auction design," *Mathematics of operations research*, vol. 6, no. 1, pp. 58–73, 1981.

[11] J. Cremer and R. P. McLean, "Full extraction of the surplus in bayesian and dominant strategy auctions," *Econometrica: Journal of the Econometric Society*, pp. 1247–1257, 1988.

[12] R. P. McAfee, J. McMillan, and P. J. Reny, "Extracting the surplus in the common-value auction," *Econometrica: Journal of the Econometric Society*, pp. 1451–1459, 1989.

[13] S. Muthukrishnan, "Ad exchanges: Research issues," in *Internet and network economics*. Springer, 2009, pp. 1–12.

[14] S. Yuan, J. Wang, B. Chen, P. Mason, and S. Seljan, "An empirical study of reserve price optimisation in real-time bidding," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2014, pp. 1897–1906.

[15] M. Ostrovsky and M. Schwarz, "Reserve prices in internet advertising auctions: A field experiment," in *Proceedings of the 12th ACM Conference on Electronic Commerce*, ser. EC '11. New York, NY, USA: ACM, 2011, pp. 59–60. [Online]. Available: http://doi.acm.org/10.1145/1993574.1993585

[16] R. Battiti, "Accelerated backpropagation learning: Two optimization methods," *Complex systems*, vol. 3, no. 4, pp. 331–342, 1989.

[17] N. Cesa-Bianchi, C. Gentile, and Y. Mansour, "Regret minimization for reserve prices in second-price auctions," in *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 2013, pp. 1190–1204.

[18] M. Mohri and A. M. Medina, "Learning algorithms for second-price auctions with reserve," *Journal of Machine Learning Research*, To Appear.

[19] M. R. Rudolph, J. G. Ellis, and D. M. Blei, "Objective variables for probabilistic revenue maximization in second-price auctions with reserve," *arXiv preprint arXiv:1506.07504*, 2015.

[20] S. A. Matthews *et al.*, *A technical primer on auction theory I: Independent private values*. Center for Mathematical Studies in Economics and Management Science, Northwestern University, 1995, vol. 1096.

[21] J. Friedman, T. Hastie, and R. Tibshirani, *The elements of statistical learning*. Springer series in statistics Springer, Berlin, 2001, vol. 1.

[22] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010*. Springer, 2010, pp. 177–186.

[23] C. Chen and O. L. Mangasarian, "Smoothing methods for convex inequalities and linear complementarity problems," *Mathematical programming*, vol. 71, no. 1, pp. 51–69, 1995.

[24] M. Kuhn and K. Johnson, *Applied predictive modeling*. Springer, 2013.

[25] L. Bottou, "Stochastic gradient descent tricks," in *Neural Networks: Tricks of the Trade*. Springer, 2012, pp. 421–436.

[26] C. Darken, J. Chang, and J. Moody, "Learning rate schedules for faster stochastic gradient search," in *Neural networks for signal processing*, vol. 2. Citeseer, 1992.

[27] N. Qian, "On the momentum term in gradient descent learning algorithms," *Neural networks*, vol. 12, no. 1, pp. 145–151, 1999.

[28] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *The Journal of Machine Learning Research*, vol. 12, pp. 2121–2159, 2011.

[29] D. Simon, *Evolutionary optimization algorithms*. John Wiley & Sons, 2013.

[30] R. Fletcher and M. J. Powell, "A rapidly convergent descent method for minimization," *The Computer Journal*, vol. 6, no. 2, pp. 163–168, 1963.

[31] B. Efron and R. J. Tibshirani, *An introduction to the bootstrap*. CRC press, 1994.

[32] R. Sakia, "The box-cox transformation technique: a review," *The statistician*, pp. 169–178, 1992.

[33] F. Munoz-Garcia, "A systematic presentation of equilibrium bidding strategies to undergraduate students," *Journal of Economics and Economic Education Research*, vol. 15, no. 2, p. 101, 2014.