Sponsored Search Auctions with Rich Ads

Ruggiero Cavallo
Yahoo Research
New York, NY
cavallo@yahoo-inc.com

Prabhakar Krishnamurthy
Yahoo Research
Sunnyvale, CA
pkmurthy@yahoo-inc.com

Maxim Sviridenko
Yahoo Research
New York, NY
sviri@yahoo-inc.com

Christopher A. Wilkens
Yahoo Research
Sunnyvale, CA
cwilkens@yahoo-inc.com

ABSTRACT

The generalized second price (GSP) auction has served as the core selling mechanism for sponsored search ads for over a decade. However, recent trends expanding the set of allowed ad formats—to include a variety of sizes, decorations, and other distinguishing features—have raised critical problems for GSP-based platforms. Alternatives such as the Vickrey-Clarke-Groves (VCG) auction raise different complications because they fundamentally change the way prices are computed. In this paper we report on our efforts to redesign a search ad selling system from the ground up in this new context, proposing a mechanism that optimizes an entire slate of ads globally and computes prices that achieve properties analogous to those held by GSP in the original, simpler setting of uniform ads. A careful algorithmic coupling of allocation-optimization and pricing-computation allows our auction to operate within the strict timing constraints inherent in real-time ad auctions. We report performance results of the auction in Yahoo's Gemini Search platform.

1. INTRODUCTION

Very early in the history of sponsored-search advertising, all the major platforms settled on some version of the Generalized Second Price (GSP) auction as the mechanism used to sell search ad spots. GSP has had remarkable staying power, apparently serving search ad marketplaces well for over a decade. However, recent trends expose problems stemming from the rigidity of traditional GSP-bound platforms: ads now come in various sizes and formats, and a mechanism that simply sorts ads and prices each based on competition from the ad below will have significant inefficiencies and unsought incentive properties.

For instance, imagine that a search platform has a priori allotted 12 lines at the top of the search page for advertisements. In the "old world" all ads were three lines long (just a title, url, and description line), and so in this 12-line example there would be precisely four available ad *slots*, regardless of which advertisers bid. But in the "new world" there may be ads that have the basic three lines *plus* additional lines of sitelinks (taking the user directly to specific sections of the advertiser's landing page), star-ratings, location information, a phone number, etc. An example of some of these ad extensions and decorations on Yahoo's search platform is given in Figure 1.

Figure 1: All highlighted sections of the above image are optional extensions to the basic three-line ad format. The search platform can include or omit them at its discretion in order to optimize the overall slate of ads presented to the user.

In the new world of heterogeneous ads, an *ad packing problem* emerges. In a context where ads can vary in length, the

WWW 2017, April 3–7, 2017, Perth, Australia.
ACM 978-1-4503-4913-0/17/04.
http://dx.doi.org/10.1145/3038912.3052703

search platform faces the richer problem of deciding which versions of which ads—and how many—to show. Whether it's best to show a larger or smaller version of an ad may depend on which size-variants of competing ads are available. Perhaps one giant ad should be chosen to fill the entire space, or perhaps it's better for the giant ad to be "trimmed" to a more moderate size and paired with a second small ad below it, or perhaps a slate of several three-line ads is best, etc. An illustration of the packing problem is given in Figure 2.

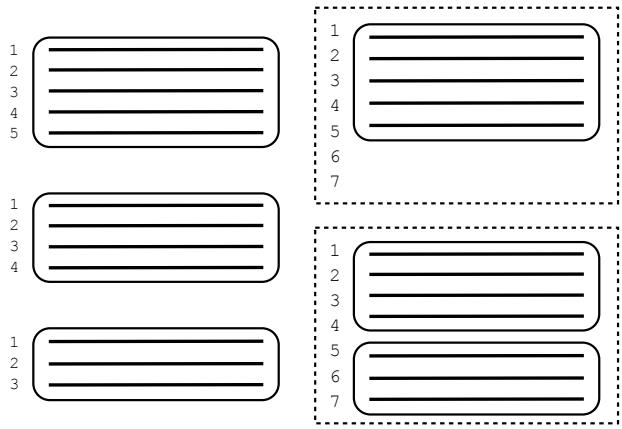

Figure 2: An illustration of the ad packing problem that arises in a context with ads of varying size. Imagine there are three ad candidates, of size 5, 4, and 3, respectively (illustrated at left), and seven lines of total available ad space. Imagine that a ranking of the ads (by whatever metric one chooses) puts them in decreasing order of size. A greedy algorithm will place the five-line ad, and then have no ability to fill the remaining two lines (top right). It may be more efficient to instead place the four- and three-line ads (bottom right).

To deal with this new world, in this paper we rethink the search ad allocation and pricing problems from the ground up, proposing a mechanism that optimizes an entire page of ads *globally*. The efficiency-maximizing ad allocation problem can be formulated as an integer program; however, for a number of reasons, solving it this way is unwieldy in practice. We instead approach the problem explicitly as a search through the space of possible ad slates. The specific solution we implement is local-search based and not guaranteed to find an optimal configuration, but in practice its distance from optimality is negligible.

The main feature of the classic approach that we retain is a separable click-probability model, wherein it is assumed that the probability that a given ad will be clicked is equal to the product of an ad-variant-specific "ad clickability" number and an ad-variant-independent "location clickability" number. Even here, though, innovations are required: the "location clickability" number can no longer be associated with an ad *slot*, since the starting-position of the i^{th} ad now depends on what kind of ad variants were shown above — location clickability for us is now a function of starting-line-position rather than slot-number.

The most technically novel contribution of this work is probably the approach we take to computing prices. It is most common to think of an allocation and pricing mechanism as proceeding in two stages — an allocation is computed, and from it (and the bids that generated it) prices are subsequently calculated. This two-phase approach makes a great deal of conceptual sense, but in our setting even fractions of a millisecond of compute-time are critical, and so a more integrated solution was required. Noting that the most salient pricing schemes can all be described in terms of the *allocation function* x (where $x_i(b_i)$ is the probability of bidder i receiving a good—here, an ad click—given that he bids b_i), during the local search phase of the algorithm we "log" key information from which we can, ultimately, very quickly compute each bidder's allocation function. Then, whatever pricing scheme is chosen, it can be applied by essentially just "reading off" prices from the computed allocation functions.

1.1 Related work

The problem of rich ads in search is well known, but not as well studied. In one sense, there is little to do — the elegant Vickrey-Clarke-Groves (VCG) auction reduces any problem related to rich ads to a modeling and optimization problem if one buys into it, and Facebook and Google have both leveraged VCG for this very reason [15].

The primary challenge for rich search ads is that marketplaces have been running GSP auctions for years; two thin lines of work consider the consequences. The first line of work studies GSP-like mechanisms in more complex optimization domains: papers by Deng et al. [5], Bacharach et al. [2], and Cavallo and Wilkens [4] generally show negative results that equilibria may be poor or nonexistent. The second line of work instead aims to convert GSP bidders to VCG bidders with minimal issues [1].

More broadly, a long line of work starting with Varian [14] and Edelman, Ostrovsky and Schwarz [8] studies GSP and attempts to rationalize its use, e.g., by showing the existence of good equilibria or showing that GSP is more robust when click-through-rates have error [6, 13]. More recently, we argue that advertisers do not have quasilinear utilities and that GSP may in fact be the truthful auction [3, 16, 17].

These prior works are all largely theoretical in nature. In the current paper, while we do make some conceptual and modeling contributions, a large emphasis is on reporting about what we think is an interesting large-scale engineering task: how to solve a computationally hard market-based problem in a feasible amount of time under severe runtime constraints. That dimension of our work strongly connects with many other studies from very different domains, such as [9, 10, 12], to name a few.

2. THE AD ALLOCATION PROBLEM

We start with a formal description of the ad allocation problem. There is a set A of ad "candidates"; each $a \in A$ has a height $h(a)$ and is associated with an advertiser $\alpha(a) \in \Lambda$, where Λ is the set of all advertisers. At most one ad per advertiser can appear on the page. There are also configurable limits on the number and cumulative height of ads that can be shown on a single page: no more than `ADLIM` ads occupying a total of `H` lines can be selected.

Each ad a has an associated bid $b_{\alpha(a)}$ and click probability density p_a. $b_{\alpha(a)}$ can be interpreted as the advertiser's claim about how much value he will receive should one of his ads be clicked (it is the same for all of his ads). The click probability density p_a is a more novel concept: it can be thought of

as a kind of normalized "click probability per line" for the ad. In combination with the line-specific location-clickability parameters,[1] it determines $p_a(k)$ for each $k \in \{0, \ldots, \mathtt{H} - 1 - h(a)\}$, which is the search platform's estimate of the probability with which a will be clicked if it is placed at starting line k.

Each ad also has an associated vector of "costs", where cost can loosely be thought of as the externality the ad imposes (on the user, the search platform, etc.) if it is shown; $c_a(k)$ denotes this cost when ad a is placed at starting line $k \in \{0, \ldots, \mathtt{H}-1-h(a)\}$. One simple way costs may be deployed in practice is to assign a constant value to every $c_a(k)$, for every a and k, using that constant as a knob with which to tune the average advertising footprint on the page.

For each ad $a \in A$ and line $j \in \{0, \ldots, \mathtt{H} - 1 - h(a)\}$, let $\mathcal{L}_{aj} = \{k : j - h(a) + 1 \leq k \leq j\}$, i.e., if an ad a starts on line $i \in \mathcal{L}_{aj}$ then the ad a covers line j.

Our goal is to maximize efficiency, i.e., total advertiser value net of costs imposed by the chosen configuration of ads. Letting $z_{a,k}$ be a boolean variable denoting whether or not ad a is placed at starting line k, we can formulate the problem as follows:

$$\text{maximize} \quad \sum_{k=0}^{\mathtt{H}-1} \sum_{a \in A} (b_a \, p_a(k) - c_a(k)) \cdot z_{a,k} \tag{1}$$

$$\text{subject to} \quad \sum_{k=0}^{\mathtt{H}-1} \sum_{a:\alpha(a)=i} z_{a,k} \leq 1, \quad \forall i \in \Lambda, \tag{2}$$

$$\sum_{a \in A} \sum_{k \in \mathcal{L}_{aj}} z_{a,k} \leq 1, \quad \forall j \in \{0, \ldots, \mathtt{H}-1\}, \tag{3}$$

$$\sum_{k=0}^{\mathtt{H}-1} \sum_{a \in A} z_{a,k} \leq \mathtt{ADLIM}, \tag{4}$$

$$z_{a,k} \in \{0, 1\}. \tag{5}$$

Constraint (2) says that we can choose at most one ad variant per advertiser. Constraint (3) says that each line can be covered by at most one ad. This constraint also implicitly encodes the fact that our solution can use at most H lines. Constraint (4) limits the total number of ads chosen by the solution.

The above is an integer program that can be solved with standard methods. Even though the problem is strongly NP-hard (by the reduction from 3-PARTITION), the number of possible ad candidates is bounded, and so asymptotic runtime analysis is really not relevant. However, the runtime constraints of this environment are extremely severe — to create an experience of "instant service" for search users, every millisecond counts, and there may not always be time to solve this integer program.

2.1 Our algorithm

Motivated by runtime constraints, we opt for a local-search based heuristic approach to the problem. Our algorithm, described below in Figure 3, virtually always obtains an optimal solution, but in a much shorter period of time; moreover, it has an "anytime" property — in the rare event of an instance that cannot be solved within our time-constraints, the local search can be shut down and the intermediate solution taken.

The algorithm starts by doing something akin to traditional GSP: it orders ads by bid times click probability—except here we use click probability *density* since ads vary in length—and then chooses a slate greedily. But while this is where traditional GSP ends, it is only a starting point for us. The core of the algorithm iteratively modifies the slate through a series of ad swaps until no improving swaps can be made. We find solutions in this way for every possible size slate, and then choose the best one.

For ad slate cardinality $K \in \{1, \ldots, \mathtt{ADLIM}\}$:

1. A **greedy** starting allocation –
   (i) Order A by bid times click-probability density.
   (ii) Select the first K ad candidates in the ordered list, iteratively reducing the set of available candidates to respect the one-ad-per-advertiser and total height constraints.
2. A **local search** loop of 1-for-1 ad swaps –
   Observe objective value X (Eq. 1).
   For each ad a in the current solution (from top to bottom):
   (i) Remove a from the slate.
   (ii) For each ad b in the set of ads that are not part of the current solution (including ad a), for each slot that b can feasibly be inserted into:
   - Insert b and observe the objective value.
   - If it exceeds X, log the swap and go to (2.).

   (iii) No swap for a improved the objective, so return a back to its original position.

   Execution for the cardinality K iteration completes when there exists no 1-for-1 ad swap that improves the objective value.

The best of the K locally optimal solutions is chosen.

Figure 3: A description of our heuristic ad allocation algorithm. In the first phase a rank-based configuration, akin to what vanilla GSP would produce, is chosen. Then in phase two it is iteratively improved until a local optimum is reached.

What are the possible vulnerabilities of this algorithm — i.e., in what cases might we get stuck in a local optimum that is not globally optimal? This may happen only in cases where swapping more than one ad at a time is required. Note that the absence of "1-for-2" swaps and the like is strongly mitigated by the fact that we find a local optimum for every possible cardinality ad slate. We will report detailed performance statistics in Section 4. For now, suffice it to say that the algorithm rarely leaves significant efficiency on the table.

[1] These may be calculated naively based on empirical click-through-rates for every line of the page. More sophisticated approaches that seek to avoid selection bias may also be applied; we do not delve into such details here.

3. PRICING

Our pricing implementation maximizes flexibility by estimating each bidder i's allocation curve x_i.[2] The allocation x_i is a common tool in theory because it fully captures what an advertiser needs to know when selecting a bid. However, auctions in practice rarely construct x_i explicitly; instead, they rely on computations that indirectly reference it. For example, externality pricing in the VCG auction is computed by removing each bidder one at a time and computing the negative effect on others — this computation happens to be equal to the area above x_i.

In our case, having direct access to x_i is important for two reasons. First and foremost, as we will discuss later, we strive to maintain GSP-like pricing, and our formulation effectively requires full knowledge of the curve x_i. Second, having access to x_i gives substantial flexibility in pricing if Yahoo wishes to change in the future, say, if competitors switch to a different pricing function such as VCG and Yahoo feels compelled to follow suit.

We will first discuss how we estimate x_i efficiently; then we will discuss a handful of possible pricing strategies and motivate GSP-like prices.

3.1 Estimating allocation curves

The local search optimization explores a wide variety of slates; we want to use these slates to efficiently construct an approximation of x_i. Since the allocation curve will be piecewise-constant,[3] our desired output is a sequence of thresholds $\tau_i[0], \ldots, \tau_i[k]$ and a sequence of allocations $\hat{x}_i[0], \ldots, \hat{x}_i[k]$, where the final estimated allocation is given by:

$$\hat{x}_i(b_i) = \begin{cases} \hat{x}_i[j] & \text{for } b_i \in [\tau_i[j-1], \tau_i[j]) \\ \hat{x}_i[k] & \text{for } b_i \geq \tau_i[k-1] \end{cases}.$$

This is conceptually easy to compute in a naïve way: identify the breakpoints τ_i by repeated binary search. Unfortunately, this will require too much time, as the allocation algorithm must be run at every stage of the binary search. We therefore instead leverage the work of local search to construct an approximation.

The approximate allocation $\hat{x}_i$.

Note that if we run the optimal algorithm, bidder i's allocation can be written as $x_i(b) = x_i(\arg\max_{\mathcal{C}} OBJ(\mathcal{C}, b))$ where $x_i(\mathcal{C})$ is the allocation probability (probability of a click) on i's ad in slate $\mathcal{C}$. Given any subset of possible slates S, we can then define an approximation $\hat{x}_i$ by taking the arg max over only those slates in S, i.e., $\hat{x}_i(b) = x_i(\arg\max_{\mathcal{C} \in S} OBJ(\mathcal{C}, b))$. We use this idea to define $\hat{x}_i$:

Definition 1. The *local search approximation* of the allocation curve is

$$\hat{x}_i(b) = x_i\left(\arg\max_{\mathcal{C} \in LS} OBJ(\mathcal{C}, b)\right),$$

where LS denotes the set of slates considered by the local search algorithm.

Note that the approximation $\hat{x}_i$ is the exact allocation assuming that the mechanism always explored a fixed set of slates and selected the optimal one.[4] However, since the mechanism will explore different slates for different bids, $\hat{x}_i$ can both over- and under-estimate x_i. The accuracy of $\hat{x}$ as an approximation of x will be discussed in Section 4.

Computing $\hat{x}_i$ efficiently.

Note that for any slate $\mathcal{C}$ and bids b, we have

$$OBJ(\mathcal{C}, b) = \sum_{i \in \Lambda} \Big(x_i(\mathcal{C}) b_i - c_i(\mathcal{C}) \Big)$$

In particular, fixing bidder i this can be written as $OBJ(\mathcal{C}) = z_{i,\mathcal{C}} + x_i(\mathcal{C}) b_i$, where

$$z_{i,C} = \sum_{j \in \Lambda \setminus \{i\}} \big(x_j(\mathcal{C}) b_j - c_j(\mathcal{C}) \big) - c_i(\mathcal{C})$$

is independent of b_i. If we let $\phi_i(b_i)$ denote the optimal objective value when i reports b_i (holding b_{-i} fixed), we can write:

$$\phi_i(b_i) = \max_{\mathcal{C}} \left\{ z_{i,\mathcal{C}} + x_i(\mathcal{C}) b_i \right\}$$

Each slate $\mathcal{C}$ yields a distinct $z_{i,\mathcal{C}} + x_i(\mathcal{C}) b_i$ (i.e., objective value as a function of b_i) line, and ϕ_i is the upper-envelope of these lines. Importantly, i's allocation when reporting b_i is the slope of the upper envelope at b_i:

Observation 1. *The optimal objective value ϕ_i, as a function of bid b_i, for a set of slates S is the upper-envelope of the lines $\{z_{i,\mathcal{C}} + x_i(\mathcal{C}) b_i\}$ associated with the slates $\mathcal{C} \in S$. The associated allocation function $\hat{x}_i$ is the slope of the upper envelope $\frac{d\phi_i}{db_i}$.*

This implies a straightforward method to compute $\hat{x}_i$, illustrated in Figure 4.

Observation 2. *The upper envelope is convex, therefore its slope is nondecreasing and the allocation $\hat{x}_i$ is nondecreasing.*[5]

3.2 Pricing methodologies

The beauty of an approach like this, which efficiently constructs an accurate representation of an entire allocation curve for each bidder, is that an array of diverse pricing functions can be accommodated—all with the same underlying infrastructure—with only a quick switch of the final "price read-off" stage (step 2 in Figure 4).

While we will ultimately settle on prices that mimic GSP, three pricing strategies are worthy of discussion here: first-pricing, VCG pricing, and GSP pricing. Each strategy has its own strengths and weaknesses.

[2] In our setting there are a variety of non-null outcomes (ranging over ad variants and the slots they may appear in) that any given bidder may receive; but an "allocation" can be reduced to the one dimension that determines advertiser value: probability of click. $x_i(b_i)$ is thus the probability with which i receives a click in the outcome yielded when he bids b_i and all other bidders' bids are held constant.

[3] Each piece corresponds to a region of the bid space that yields the same allocation.

[4] Said in terms of another standard mechanism, $\hat{x}_i$ is the allocation of a maximal-in-range allocation on the set of slates LS.

[5] This should not be confused with a claim that x_i is nondecreasing; if the local search fails to consider the right set of possible slates, its suboptimalities may lead to nonmonotonicities in the actual allocation function x. $\hat{x}_i$ remains monotonic by construction.

Every time the objective value of a feasible slate $\mathcal{C}$ is computed in local search, store $x_i(\mathcal{C})$ and $z_{i,C} = \sum_{j \in \Lambda \setminus \{i\}} \big(x_j(\mathcal{C})b_j - c_j(\mathcal{C})\big) - c_i(\mathcal{C})$ for each bidder i.

After local search terminates:

1. For each bidder i, compute $\hat{\phi}_i$ as the upper envelope of the lines:

   $$\{z_{i,\mathcal{C}} + x_i(\mathcal{C})b_i\}$$

   This gives a piecewise linear function $\hat{\phi}_i$ composed of lines (in order):

   $$(z_i[0], \hat{x}_i[0]), (z_i[1], \hat{x}_i[1]), \ldots, (z_i[k], \hat{x}_i[k]),$$

   with inflection points:

   $$\tau_i[0] = 0, \tau_i[1], \ldots, \tau_i[k] \ .$$

2. Read off the (stepped) allocation curves as the derivative of the upper envelope:

   $$\hat{x}_i(b_i) = \begin{cases} \hat{x}_i[j] & \text{for } b_i \in [\tau_i[j], \tau_i[j+1]) \\ \hat{x}_i[k] & \text{for } b_i \geq \tau_i[k] \end{cases}$$

Figure 4: Algorithm for constructing an estimated allocation curve $\hat{x}_i$.

First-pricing.

First-price auctions (advertisers pay exactly what they bid) are convenient to implement but create major issues. Simple implementations are proven to be unstable both in theory and in practice [7]. While stability can be restored [11], bidders must adopt a new bidding language. Perhaps more damningly, first-price semantics would likely upset advertisers who are generally accustomed to a second-price discount on search.

VCG pricing.

Running a traditional Vickrey-Clarke-Groves (VCG) auction is appealing for many reasons, but is ultimately an unsatisfactory solution. On the plus side, first, standard theory says that it is the truthful auction. Second, VCG prices can be efficiently computed as externalities — it is sufficient to rerun the optimization as a black box n additional times, then compute the negative effect each bidder has on the others. This mathematical abstraction naturally leads to a practical implementation abstraction, making VCG prices easy to implement. As a result, VCG has become the industry standard auction when facing a complex optimization problem [15].

However, VCG is not a perfect solution. Practically speaking, the marketplaces that use VCG pricing have generally done so from an early stage — we are unaware of any mature markets that have *transitioned* from GSP to VCG. The main challenge is that advertisers will need to change their bidding strategies; until they do, the auctioneer will generally lose money. Even assuming bidders eventually react, obtaining a smooth transition is a tricky task [1]. Even worse in our particular circumstance, it is unclear that advertisers will be responsive given Yahoo's market share.

More subtly, it is not clear that VCG is truly the best auction from a theoretical viewpoint for reasons having to do with questions regarding which utility model best reflects advertiser preferences. In particular, our prior work even suggests that GSP might be the appropriate incentive compatible auction [3, 17].

Generalized GSP pricing.

A natural solution is to stay with GSP pricing; the challenge is to define what that means. A traditional GSP auction sorts ads by a ranking score and charges each bidder the minimum bid required to hold its rank. This is sensible when the auction is simply assigning ads to ranks; but when the auction makes a complex trade-off over the features of an ad, this is no longer well-defined. A theoretical literature strives to justify GSP's use; however, it fails to identify the defining properties of GSP that one would need in order to generalize it.

Based on our prior work, we propose that GSP be generalized as the truthful auction for value maximizing bidders (see [3, 16, 17] for a thorough treatment). A value maximizing bidder wants to get as many clicks as possible without paying more than its value, i.e., to maximize x_i while keeping $p_i \leq v_i$. In contrast, a traditional model assumes bidders maximize expected profit $(v_i - p_i)x_i$.

Defining truthful prices for these bidders in our auction leads to a pricing intuition often given to the GSP price:[6]

Definition 2. The *truthful price p_i for a value maximizer* is $p_i[j] = \tau_i[j]$ when i gets allocation $x_i[j]$.

That is, p_i is the minimum bid advertiser i must submit to maintain the same allocation.

This gives us a candidate auction: when i gets allocation $x_i[j]$, charge $p_i[j] = \tau_i[j]$. Unfortunately, this auction may "overcharge." For example, if $x_i[j] \approx x_i[j-1]$ (there's a tiny step in the allocation function) but $\tau_i[j] \geq 2\tau_i[j-1]$ (there's a large difference in the minimum bids that yield the two allocations), an advertiser might not care whether it gets allocation $x_i[j]$ or $x_i[j-1]$, but this GSP auction could charge a 2x premium for the higher allocation. This is illustrated in Figure 5. The problem arises because the value maximizing model assumes bidders are willing to pay an unrealistically large price for a tiny increase in allocation.

To refine our version of GSP, we choose a middle ground between VCG pricing and GSP pricing using ideas introduced in [3] and developed in the Appendix of the current paper. Our approach is to start with a hybrid preference model — a model of bidder preferences that lies between quasilinear utilities and value maximizing preferences — and set prices so that bidders of the chosen type would be truthful. We propose two different hybrids.

Our first hybrid model adds a return on investment (ROI) constraint of α to existing quasilinear utilities. We refer the reader to the Appendix for details, but the prices are as follows:

Definition 3. The *ROI-constrained truthful price* $p_i[j]$ when i gets allocation $x_i[j]$ is computed by the following recursive formula: $p_i[0] = 0$, and for all $j > 0$,

$$p_i[j] = \min\left(\tau_i[j], \frac{\hat{x}_i[j-1]p_i[j-1] + (\hat{x}_i[j] - \hat{x}_i[j-1])(\alpha+1)\tau_i[j]}{\hat{x}_i[j]}\right) \tag{6}$$

[6]Observing this property of GSP prices is not new, but [3] is the first to give a solid foundation for why this property is significant.

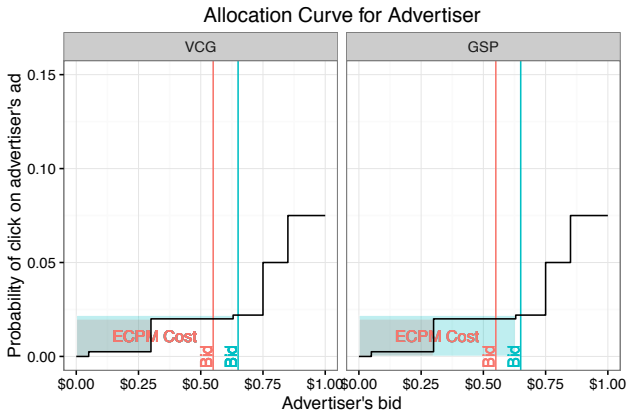

Figure 5: An illustration of "overcharging" under generalized GSP payments. The red and blue shaded areas represent the prices charged for the red and blue bids respectively, under VCG (left) and generalized GSP (right) prices. Note that the small increase in click probability results in a small change in the VCG price but a large increase in the generalized GSP price.

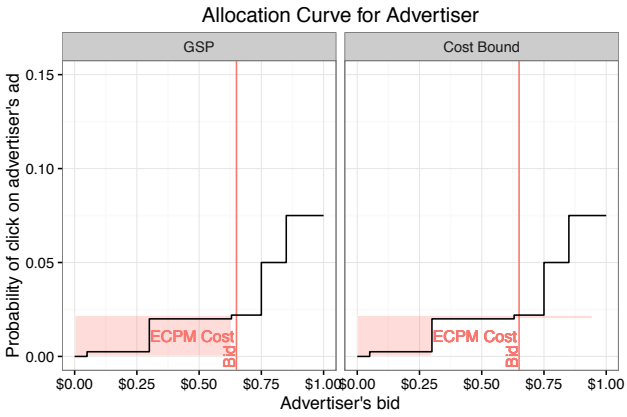

Figure 6: An illustration of pricing for ROI-constrained bidders. Pricing for bidders with an ROI constraint charges the smaller of two quantities: the GSP price (left) and a maximum-marginal-cost-per-click increase over the previous price (right).

This formula has a natural interpretation: a bidder is charged the lesser of the GSP price ($\tau_i[j]$) and the price one computes starting with $x_i[j-1]$ at price $p_i[j-1]$ and assuming a marginal cost-per-click of $(\alpha+1)\tau_i[j]$ for the extra $x_i[j]-x_i[j-1]$ expected clicks. This is illustrated in Figure 6.

The second type of preferences we propose is continuous and assumes that bidders optimize a utility function of the form $u_i = v_i^{\alpha+1} - p_i^{\alpha+1}$. Again, we refer the reader to the Appendix for details:

Definition 4. The *α-hybrid truthful price* $p_i[j]$ is given by the following formula:

$$p_i[j] = \frac{1}{\hat{x}_i[j]}\left(\sum_{l=1}^{j}(\tau_i[l]\hat{x}_i[l])^{\alpha+1} - (\tau_i[l]\hat{x}_i[l-1])^{\alpha+1}\right)^{\frac{1}{\alpha+1}} \quad (7)$$

At $\alpha = 0$, both models describe traditional VCG prices (truthful prices for profit maximizers); as $\alpha \to \infty$, both models converge to GSP prices (truthful for value maximizers). We choose a hybrid model to mimic GSP while curtailing extremely high marginal prices.

4. RESULTS

4.1 Allocation accuracy

The first results we present regard how well our heuristic ad allocation algorithm approximates the optimal solution.[7] We report results on a random selection of 100,000 auction instances drawn from Yahoo's Gemini search platform for Desktop devices. An "instance" consists of a set of candidate ads and all relevant accompanying information (bids, clickability predictions, size, decorations, etc.). We ran the algorithm with a variety of different maximum ad cardinality limits (`ADLIM`), in each case applying a maximum number of ad lines (`H`) equal to 18.

`ADLIM`	2	3	4	5
Efficiency rate	0.9999	0.9998	0.9978	0.9957
Optimality rate	0.999	0.993	0.877	0.806

Table 1: Performance comparison of our heuristic algorithm against the optimal algorithm. The *efficiency rate* is the average ratio of our algorithm's efficiency to that of the optimal algorithm; the *optimality rate* is the percentage of instances on which our heuristic returned a globally optimal solution (i.e., achieved *efficiency rate* of 1).

As Table 1 indicates, when a maximum of two ads may be shown, the heuristic misses the optimal allocation in less than one out of every 10,000 instances. When three ads may be shown, this goes down to about one out of every 150 instances. As the `ADLIM` increases the heuristic diverges from the optimal solution in more and more cases; however, when it does diverge, it still finds a solution that is negligibly worse than the optimal one. Even for an `ADLIM` of 5 (which is the upper limit of what is currently seen on any of the major search platforms), our heuristic algorithm obtains more than 99.5% of the optimal efficiency on average.

4.2 Pricing accuracy

Completely apart from the potential suboptimality of the allocation that our algorithm computes, there is approximation in the prices we compute. As discussed, to precisely compute prices (say, according to Eq. (6) or Eq. (7)) one needs to compute the *allocation curve* for the bidder, from which the price can be quickly deduced. One can do so in a brute-force manner, panning across the space of possible bids and observing how the bidder's allocation (and

[7]The optimal allocation, given any set of ad candidates, can be solved by a variety of methods including integer programming; it is easy for us to compute statistics about an optimal algorithm offline, despite it not being suitable for online use due to the severe runtime constraints of our domain.

predicted number of clicks) changes, holding all other bids constant. Since there are only a finite number of allocations, one can do better than this by using binary search to determine the "break points" in the allocation curve — i.e., the set of distinct flat regions it is constituted by. But even this will be too computationally costly to use in real-time. Hence our *online* method for computing allocation curves, described in Section 3.

Determining those prices is computationally feasible, but are the prices any good? Yes. Figure 7 illustrates the accuracy of our "approximate prices" by comparing them to the exact prices, computed offline via binary search.

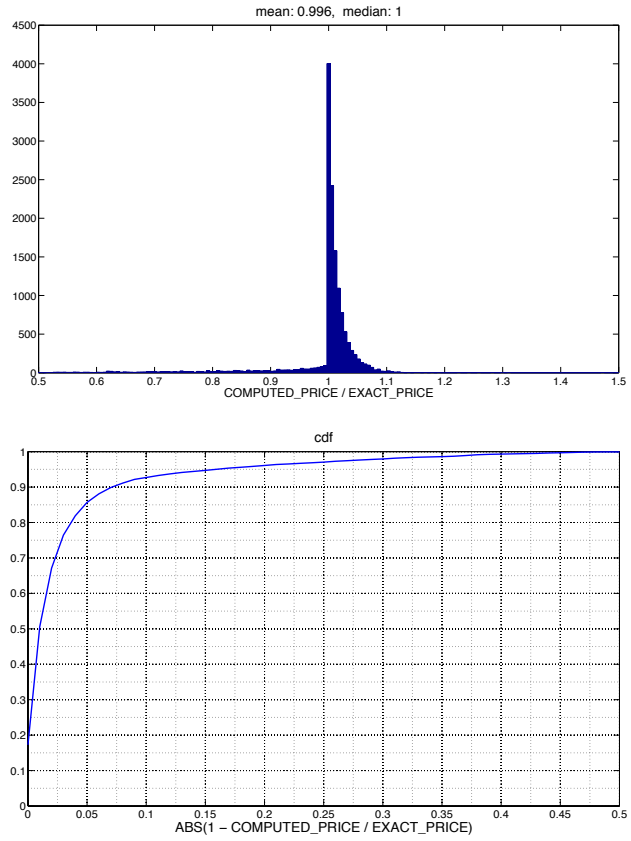

Figure 7: The distribution of the ratio of our computed prices to exact prices (top), and the cdf of the distance between our estimate and the exact price (bottom).

The top illustration in Figure 7 is a histogram of the ratios of approximated price to exact price. The giant peak at 1 indicates that we get the price precisely right a significant percentage of the time. There is a large volume just to the right of peak, where we slightly overestimate prices, and there is a fairly long and light tail below 1. The bottom illustration conveys the cumulative distribution of the absolute-value distance of our approximate prices from the exact prices. We're within 5% of the exact price about 85% of the time, and we're within 15% almost 95% of the time.

4.3 Online bucket results

In live bucket tests our algorithm substantially improves over its predecessor, at least along some metrics, packing more ads into less space. Selected results from one test are given in Table 2: the test packed $> 10\%$ more ads into $< 90\%$ of the lines. Meanwhile, it increased advertiser value by 8% (assuming bids are truthful) with clicks remaining nearly neutral. Revenue was also neutral, but this metric has little meaning in a small bucket test when the auction rules change.

Metric	Test vs. Control
Total Ads	+11%
Total Lines	−11%
Value per Search	+8%
Revenue per Search	neutral
Click Yield	−1%

Table 2: Selected results from a bucket test on Yahoo's desktop search platform. The test ran for 5 days on 1.5% of traffic.

5. DISCUSSION

In this paper we provided a formal introduction of the *rich ads* problem for sponsored search, and described our recent efforts to address it. The major part of the paper focused on presenting the details of our engineered solution. We described a local search based heuristic method that achieves performance that is practically identical to that of an optimal algorithm, while meeting the tight runtime constraints of the sponsored search domain. We described a method that couples the algorithmic determination of a near-optimal allocation with allocation-curve construction, which allows us to quickly compute prices without repeating work, making the whole system runtime feasible and easily adaptable to future developments.

The claim about our heuristic optimization algorithm being near-optimal is an empirical observation based on the types of decorations available today and the ad real-estate constraint in place. Similarly we have empirically established the close approximation of our allocation curve generation method. In our future work, we plan to explore exact optimization algorithms that are guaranteed to produce allocation curves with approximation bounds.

Online experiments on a fraction of Yahoo search traffic comparing the performance of our algorithm with the standard GSP algorithm on metrics such as revenue, click yield, ad real estate footprint, and user response indicate that our new approach yields improved outcomes in a "win-win-win" fashion, achieving gains in advertiser value and revenue to the search platform, while at the same time reducing the overall ad footprint, which is presumably in the user's best interest. One thing we observe is that, for many ads, after a certain point the click-through-rate (CTR) per line has diminishing returns and thus smaller ads have a higher average click-through-rate per line. Our algorithm therefore often favors smaller ad variants over larger ones, packing more ads for the same total number of lines on a page.

After a version of our algorithm is launched into full-scale production we expect that advertisers will adjust their bids in an effort to have their preferred ad variant appear. While in our present version we assume that we can drop decorations willy-nilly to vary the size of the ad, advertiser preferences over their ad variants may present practical con-

straints. It may be necessary to allow advertisers to bid separately for each ad variant, so that their preferences can be properly expressed. This would raise a number of challenges, among other things forcing us to modify how we generate allocation curves for pricing. It is an area that we will be studying further.

6. REFERENCES

[1] Yoram Bachrach, Sofia Ceppi, Ian A. Kash, Peter Key, and Mohammad Reza Khani. Mechanism design for mixed ads. In *11th Workshop on Sponsored Search Auctions*, 2015.

[2] Yoram Bachrach, Sofia Ceppi, Ian A. Kash, Peter Key, and David Kurokawa. Optimising trade-offs among stakeholders in ad auctions. In *Proceedings of the 15th ACM Conference on Economics and Computation (EC'14)*, pages 75–92, 2014.

[3] Ruggiero Cavallo, Prabhakar Krishnamurthy, and Christopher A. Wilkens. On the truthfulness of GSP. In *11th Workshop on Sponsored Search Auctions*, 2015.

[4] Ruggiero Cavallo and Christopher A. Wilkens. GSP with general independent click-through-rates. In *Proceedings of the 10th International Conference on Web and Internet Economics (WINE'14)*, pages 400–416, 2014.

[5] Xiaotie Deng, Yang Sun, Ming Yin, and Yunhong Zhou. Mechanism design for multi-slot ads auction in sponsored search markets. In *Proceedings of the 4th International Workshop on Frontiers in Algorithmics (FAW'10)*, pages 11–22, Berlin, Heidelberg, 2010.

[6] Paul Dütting, Felix Fischer, and David C. Parkes. Truthful outcomes from non-truthful position auctions. In *Proceedings of the 17th ACM Conference on Economics and Computation (EC'16)*, pages 813–813, New York, NY, USA, 2016. ACM.

[7] Benjamin Edelman and Michael Ostrovsky. Strategic bidder behavior in sponsored search auctions. *Decision Support Systems*, 43(1):192–198, February 2007.

[8] Benjamin Edelman, Michael Ostrovsky, and Michael Schwarz. Internet advertising and the generalized second-price auction: Selling billions of dollars worth of keywords. *American Economic Review*, 97(1):242–259, 2007.

[9] Yuzo Fujishima, Kevin Leyton-Brown, and Yoav Shoham. Taming the computational complexity of combinatorial auctions: Optimal and approximate approaches. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI'99)*, pages 548–553, 1999.

[10] Oktay Günlük, László Ladányi, and Sven De Vries. A branch-and-price algorithm and new test problems for spectrum auctions. *Management Science*, 51(3):391–406, 2005.

[11] Darrell Hoy, Kamal Jain, and Christopher A. Wilkens. A dynamic axiomatic approach to first-price auctions. In *Proceedings of the Fourteenth ACM Conference on Electronic Commerce (EC'13)*, pages 583–584, New York, NY, USA, 2013. ACM.

[12] Benjamin Lubin, Adam I. Juda, Ruggiero Cavallo, Sébastien Lahaie, Jeffrey Shneidman, and David C. Parkes. ICE: An expressive iterative combinatorial exchange. *Journal of Artificial Intelligence Research*, 33(1):33–77, 2008.

[13] Paul Milgrom. Simplified mechanisms with an application to sponsored-search auctions. *Games and Economic Behavior*, 70(1):62–70, 2010.

[14] Hal R. Varian. Position auctions. *International Journal of Industrial Organization*, 25:1163–1178, 2007.

[15] Hal R. Varian and Christopher Harris. The VCG auction in theory and practice. *American Economic Review*, 104(5):442–45, 2014.

[16] Christopher A. Wilkens, Ruggiero Cavallo, and Rad Niazadeh. Mechanism design for value maximizers. *CoRR*, abs/1607.04362, 2016.

[17] Christopher A. Wilkens, Ruggiero Cavallo, and Rad Niazadeh. GSP — the cinderella of mechanism design. In *Proceedings of the 26th International Conference on World Wide Web (WWW'17)*, 2017.

APPENDIX

A. TRUTHFUL AUCTIONS FOR GENERAL PREFERENCES

In Section 3 we proposed two different pricing schemes that hybridize between VCG and GSP pricing. Each scheme is derived by hypothesizing a family of bidder preferences for which VCG is truthful at one extreme and GSP is truthful at another. To define truthful prices for these preferences we introduce *indifference point pricing*. Throughout this section we focus on the perspective of a single bidder i and therefore drop its subscripts.

Definition 5. (Indifference Point Auction) Let $v \in \Re^+$ denote the type of a bidder and $u(x, v, p)$ denote its utility for getting x expected clicks at price-per-click p. The *indifference point prices* $p[j]$ for a monotone discrete allocation curve $\{(\tau, \hat{x})\}$ are computed recursively for a continuous utility function u by setting $p[0] = 0$ and then choosing $p[j]$ to satisfy:

$$u(\hat{x}[j], \tau[j], p[j]) = u(\hat{x}[j-1], \tau[j], p[j-1])$$

I.e., a bidder with type $\tau[j]$ is indifferent between getting $\hat{x}[j-1]$ clicks at price $p[j-1]$ and getting $\hat{x}[j]$ clicks at price $p[j]$. More generally, when u is discontinuous, we take:

$$p_i = \inf \{\pi \,|u(\hat{x}[j], \tau[j], z) \leq u(\hat{x}[j-1], \tau[j], p[j-1])\} \quad .$$

It is easy to verify that these prices coincide with Myerson's formula when u represents traditional quasilinear utilities.

We will prove a statement about truthfulness later, but for the purposes of our paper, we use the following observations that follow by examining the indifference points of a bidder:

Observation 3. *Suppose* $u(x, v, p) = (xv)^{\alpha+1} - (xp)^{\alpha+1}$ *for* $\alpha \geq 0$. *Then the indifference point prices will be:*

$$p[j] = \frac{1}{\hat{x}[j]} \left(\sum_{l=1}^{j} (\hat{x}[l]\tau[l])^{\alpha+1} - (\hat{x}[l-1]\tau[l])^{\alpha+1} \right)^{\frac{1}{\alpha+1}}$$

Proof. Indifference of a bidder with type $\tau[j]$ implies:

$$\begin{aligned} &(\hat{x}[j]\tau[j])^{\alpha+1} - (\hat{x}[j]p[j])^{\alpha+1} \\ &= (\hat{x}[j-1]\tau[j])^{\alpha+1} - (\hat{x}[j-1]p[j-1])^{\alpha+1} \end{aligned}$$

And thus,

$$\begin{aligned}&(\hat{x}[j]p[j])^{\alpha+1}\\ &= (\hat{x}[j]\tau[j])^{\alpha+1} - (\hat{x}[j-1]\tau[j])^{\alpha+1} + (\hat{x}[j-1]p[j-1])^{\alpha+1}\\ &= \sum_{l=1}^{j}(\hat{x}[l]\tau[l])^{\alpha+1} - (\hat{x}[l-1]\tau[l])^{\alpha+1},\end{aligned}$$

from which the observation follows. □

OBSERVATION 4. *Suppose that a bidder requires an ROI of at least α, but that her utilities are otherwise quasilinear. Then if v represents what she is willing to pay (so $(1+\alpha)v$ is her true value), then we can write a utility function:*

$$u(x,v,p) = \begin{cases}(1+\alpha)xv - xp & \text{if } v \geq p\\ xv - xp & \text{otherwise.}\end{cases}$$

Indifference point prices are given by the recursion:

$$p[j] = \min\left(\tau[j], \frac{\hat{x}[j-1]p[j-1] + (\hat{x}[j]-\hat{x}[j-1])(1+\alpha)\tau[j]}{\hat{x}[j]}\right)$$

PROOF. Consider the "indifference" of a bidder with type $\tau[j]$. Suppose we are given $p[j-1]$. If we wishfully suppose that $\tau[j] \geq p[j]$, then indifference implies

$$\hat{x}[j]((1+\alpha)\tau[j] - p[j]) = \hat{x}[j-1]((1+\alpha)\tau[j] - p[j-1]),$$

and thus

$$p[j] = \frac{\hat{x}[j-1]p[j-1] + (\hat{x}[j]-\hat{x}[j-1])(1+\alpha)\tau[j]}{\hat{x}[j]},$$

which is the second term in the min.

Suppose that our wish was incorrect, i.e., suppose

$$z = \frac{\hat{x}[j-1]p[j-1] + (\hat{x}[j]-\hat{x}[j-1])(1+\alpha)\tau[j]}{\hat{x}[j]} > \tau[j] \ .$$

Then for any $p[j] \leq \tau[j]$ it must be that $u(x[j],\tau[j],p[j]) > u(x[j-1],\tau[j],p[j-1])$. On the other hand, for any $p[j] > \tau[j]$ we have $u(x[j],\tau[j],p[j]) < 0 \leq u(x[j-1],\tau[j],p[j-1])$. This implies that

$$\begin{aligned}p[j] &= \inf\{\pi \,|u(\hat{x}[j],\tau[j],z) \leq u(\hat{x}[j-1],\tau[j],p[j-1])\}\\ &= \tau[j]\end{aligned}$$

□

Our theorem about the truthfulness of indifference point pricing uses standard techniques to show that truthfulness is almost always a best response:

THEOREM 1. *Let $v \in \Re^+$ denote the type of a bidder and $u(x,v,p)$ denote his utility for getting x expected clicks at price-per-click p satisfying the following properties:*

1. *u is strictly increasing in x and v and strictly decreasing in p.*
2. *$u(x+dx, v+dv, p+dp) - u(x, v+dv, p) > u(x+dx, v, p+dp) - u(x,v,p)$ where $dx, dv, dp > 0$.*

For the indifference point auction, truthful bidding is always a best response if u is continuous or $v \neq \tau[j]$ for all j.

PROOF. Suppose that a bidder who reports v gets $x(v) = \hat{x}[j]$. First we argue that a bidder of type v never wants to report $b < v$. Fix a bid $b < v$ that results in allocation $x(b) = \hat{x}[j'] < \hat{x}[j] = x(v)$ (note $j' < j$ by construction). Let $b' \leq v$ be such that $x(b') = \hat{x}[j'+1]$. Then by monotonicity of u and the definition of p we know that a bidder of type b' does not prefer to lie and say b:

$$u(\hat{x}[j'+1], b', p[j'+1]) - u(\hat{x}[j'], b', p[j']) > 0$$

Then, since $v \geq b'$, we can use the conditions of the theorem to conclude that

$$\begin{aligned}&u(\hat{x}[j'+1], v, p[j'+1]) - u(\hat{x}[j'], v, p[j'])\\ &\geq u(\hat{x}[j'+1], b', p[j'+1]) - u(\hat{x}[j'], b', p[j'])\\ &> 0,\end{aligned}$$

and thus a bidder of type v would not lie and say b because a lie to b' (yielding $\hat{x}[j'+1]$ clicks at $p[j'+1]$) would generate more utility.

Second, we argue that a bidder does not want to report $b > v$ where $x(b) = \hat{x}[j'] > \hat{x}[j] = x(v)$. Let $b' \geq v$ be such that $x(b') = \hat{x}[j'-1]$. Suppose that we pick b' so that $b' \neq \tau[j']$. We know by definition of p that a bidder of type b' does not prefer to lie and say b:

$$u(\hat{x}[j'], b', p[j']) - u(\hat{x}[j'-1], b', p[j'-1]) < 0$$

And, therefore, by the conditions of the theorem:

$$\begin{aligned}&u(\hat{x}[j'], v, p[j']) - u(\hat{x}[j'-1], v, p[j'-1])\\ &\leq u(\hat{x}[j'], b', p[j']) - u(\hat{x}[j'-1], b', p[j'-1])\\ &< 0\end{aligned}$$

So a bidder of type v would not lie and say b because a lie to b' would generate more utility. In the case where the only valid b' has $b' = \tau[j'-1]$ we can conclude that $b' = v = \tau[j'-1]$ and so the theorem need not hold (though it is straightforward to argue that it still holds if u is continuous). □

COROLLARY 1. *Indifference point pricing is truthful "almost everywhere" (see [17]), i.e., as long as $v \neq \tau[j]$ for all j. This can be strengthened to ordinary truthfulness when u is continuous.*