Multiplicative Bidding in Online Advertising

MOHAMMADHOSSEIN BATENI, Google Research, New York
JON FELDMAN, Google Research, New York
VAHAB MIRROKNI, Google Research, New York
SAM CHIU-WAI WONG, University of California at Berkeley

In this paper, we initiate the study of the multiplicative bidding language adopted by major Internet search companies. In multiplicative bidding, the effective bid on a particular search auction is the product of a base bid and bid adjustments that are dependent on features of the search (for example, the geographic location of the user, or the platform on which the search is conducted). We consider the task faced by the advertiser when setting these bid adjustments, and establish a foundational optimization problem that captures the core difficulty of bidding under this language. We give matching algorithmic and approximation hardness results for this problem; these results are against an information-theoretic bound, and thus have implications on the power of the multiplicative bidding language itself. Inspired by empirical studies of search engine price data, we then codify the relevant restrictions of the problem, and give further algorithmic and hardness results. Our main technical contribution is an $O(\log n)$-approximation for the case of multiplicative prices and monotone values. We also provide empirical validations of our problem restrictions, and test our algorithms on real data against natural benchmarks. Our experiments show that they perform favorably compared with the baseline.

Categories and Subject Descriptors: F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Computations on discrete structures

Additional Key Words and Phrases: Approximation algorithms; multiplicative bidding

1. INTRODUCTION

Real-time ad auctions play a vital role in monetizing the Internet. In a real-time ad auction, bids are entered by the advertiser beforehand, and the auction is conducted at the time of a pageview or search. Each individual search query or pageview has a possibly unique set of features (e.g., geographic location, time of day, device, etc.) that can have a significant effect on the value of the ad to the bidder, as well as the market price of the ad placement.

Recently, some of the major search engines have started to allow an advertiser to set *bid adjustments* on their ad campaign in order to account for differences in valuation that are a function of these types of features [Google Support 2014; Bing Ads 2014]. Indeed the transition to this mode of bidding has been characterized as one of the most important recent changes to AdWords [HubSpot 2013]. For example, one could set a bid adjustment of 1.1 for search queries originating in California, an adjustment of 0.9 for queries submitted from 2-3pm, and another adjustment of 1.2 for mobile devices. Then, for a base bid of $\$1$, your final bid on a California mobile query between 2-3pm would be $\$1 \times 1.1 \times .9 \times 1.2 = \$1.188$.

Bid adjustments allow an advertiser to express relative valuation across a supported feature type (for example, geographic location), but do not allow for specifying valuation on arbitrary combinations of features. For example, if an advertiser found that

Work done while S. Wong was at Google Research, New York.

EC'14, June 8–12, 2014, Stanford, CA, USA. ACM 978-1-4503-2565-3/14/06.
http://dx.doi.org/10.1145/http://dx.doi.org/10.1145/2600057.2602874

mobile searches were 30% more valuable than desktop searches in New York, but only 15% more valuable in California, then this would not be expressible in the language of bid adjustments. Such limitations are inevitable, as the space of possible combinations of these features is prohibitively large. There are, of course, other bidding schemes with succinct bid representations. We have chosen to study multiplicative bidding here because it is the status quo. Investigating the expressive power of other schemes is an interesting future direction.

In this paper, we initiate the theoretical and empirical study of multiplicative bidding and examine the task faced by advertisers given the option of setting bid adjustments. We begin by whittling it down to a simple, elegant optimization problem on two feature dimensions. This problem still captures the tension of bidding "multiplicatively," rather than individually on each auction, while ignoring some of the unrelated idiosyncrasies of search ad auctions. We then fully explore the complexity of this optimization problem. We first show that the problem is $\Omega(\sqrt{n})$-hard to approximate, and give an algorithm which exactly matches this hardness ratio. Because these results are in relation to a solution where a bidder can bid individually on each feature combination, this also implies an information-theoretic limitation of the multiplicative bidding language itself.

Motivated by analyses on real search auction data, we then examine the effect of assuming various conditions (e.g., monotonicity on the values and prices in different dimensions). As our main technical contribution, we develop an $O(\log n)$ approximation algorithm when prices are multiplicative and values are monotone in one dimension. We validate our assumptions on search auction data, and test our algorithm on this data against natural benchmarks. Before elaborating on our results and techniques, we present a formal model of the multiplicative bidding problem, and later (in Section 1.2) describe details of our theoretical and empirical results.

1.1. Model

Most search engines conduct some variant of the *generalized second price (GSP)* auction to sell ad placements on user search queries. Since we are studying the advertiser-facing budget optimization problem, an appropriate model for an individual auction would be the "landscape" model from Feldman et al. [2007] (see also Section 1.3). In this model there is a set of *threshold* bids $b_1, \ldots, b_n$ (where n is the number of positions on the page, a small constant), and bidding in the interval $[b_i, b_{i+1}]$ gives some number of clicks at a cost of b_i per click. A special case of this model (when $n = 1$) is a take-it-or-leave-it click at a fixed price p. We will assume this special case in the present paper for simplicity, since the task of multiplicative bidding is still sufficiently sophisticated in this case. Extending to multiple click supply, multiple ad positions, or multiple queries with different market prices is an interesting direction for future work.

The bidding dimensions supported by the major search engines include time of day, geographic location, platform (e.g., mobile device vs. desktop) and keyword targeting. Thus it is reasonable to assume that the number of dimensions is small, but the number of different values in each dimension is possibly large (for example, AdWords allows bid multipliers at 15-minute intervals in a week, and at postal-code geographic level). A reasonable place to start is the 2-dimensional problem, where each dimension has a large number of possible values; for the purposes of technical focus, we consider only this case in the paper, and leave it open to extend our results to multiple dimensions.

Given these modeling considerations, we now propose the first model for the multiplicative bidding problem. We feel that this simple model retains the salient feature of the problem—namely, the tension of bidding multiplicatively—and is a solid foundation on which to inspire future work in this area.

The Multiplicative Bidding Problem. Suppose there are two *bid adjustment dimensions* with m and n possible settings, respectively. For each *entry* $(i, j) \in [m] \times [n]$, an advertiser is given a price $p_{ij} > 0$ and value $v_{ij} \geq 0$. He is required to specify a bid multiplier r_i for each row $i \in [m]$ and c_j for each column $j \in [n]$. The effective bid for cell (i, j) is then $r_i \cdot c_j$.

A cell (i, j) is said to be *captured* if its effective bid is at least the price, i.e., $r_i \cdot c_j \geq p_{ij}$. The advertiser also has a budget $B > 0$ and is subject to the budget constraint

$$\sum_{(i,j): r_i c_j \geq p_{ij}} p_{ij} \leq B.$$

His objective is to maximize the total value gained

$$\sum_{(i,j): r_i c_j \geq p_{ij}} v_{ij}.$$

Relation to knapsack. Multiplicative bidding can be viewed as a restricted version of the classical knapsack problem. Indeed, if we were free to bid any amount on each individual cell, we would be able to capture any desired subset of the cells, where each cell is simply an item with a price and value. We will refer to this solution as the *individual bidding optimum*, or simply, OPT.

With multiplicative bidding, as we shall see, not all subsets of the cells can be captured. Consequently, we are optimizing over a smaller space, and the best solution available can be no better than the individual bidding optimum.

Approximation benchmarks. One of our objectives is to quantify how much efficiency is potentially lost by restricting an advertiser to multiplicative bidding (compared to a real-time bid, as is common in Ad Exchanges, for example). In light of this, the most natural benchmark would be the optimal individual bidding solution OPT. We abuse notation by using OPT to denote both the set of cells in the optimum as well as their total value.

To simplify our presentation, we will assume that OPT simply chooses the cells with the best v_{ij}/p_{ij} ratio while not exhausting the budget B. This assumption is reasonable in two ways. First of all, the reader may have already noticed that the above is the well-known 2-approximation for knapsack and thus we lose a factor of at most 2 by adopting such a solution. Secondly, in the context of Internet advertising, very often each p_{ij} and v_{ij} is small compared to the overall spend and value derived, in which case our solution is in fact $(1 - \epsilon)$-approximate. More specifically, we assume that $p_{ij}/B < \epsilon$ for all (i, j). The reader may readily verify that this is indeed an $(1 - \epsilon)$-approximation. We stress again that almost all our results remain valid without this assumption since OPT is a 2-approximation to the true optimum. This assumption is merely introduced in order to simplify our exposition and avoid being overly verbose.

Another more benign benchmark would be the multiplicative bidding optimum, which is useful for characterizing the computational hardness of finding a good solution. This is different from the former benchmark which carries the flavor of "information theoretical" lower bounds. Somewhat surprisingly, at least in the general case, the optimal approximation ratio is essentially the same with respect to the individual bidding optimum (Lemma 2.1) and the multiplicative bidding optimum (Lemma 2.4).

1.2. Our contributions and techniques

The most important goal of our work is to establish a foundational multiplicative bidding problem on which to build algorithmic insight. Given the prevalence of this new bidding language, this represents an urgent call to investigate these bidding schemes in different scenarios and to design better bid optimization algorithms for them.

Table I. Lower bounds and algorithmic results on the approximation ratio in different cases.

	General	Monotone value- -over-price	Monotone prices and values	Multiplicative prices	Multiplicative prices and monotone values
Hardness	$\Omega(\sqrt{n})$ [1] Lemma 2.1	1	$\Omega(n^{1/2-\epsilon})$ Lemma 2.3	$\Omega(\sqrt{n})$ Lemma 2.1	1
Algorithm	$O(\sqrt{n})$ Theorem 3.3	1 Corollary 4.3	$O(\sqrt{n})$ Theorem 3.3	$O(\sqrt{n})$ Theorem 3.3	$O(\log n)$ Theorem 5.6

To this end, we first start with the plain formulation of the problem as stated in Section 1.1, and we are able to fully characterize its approximability to be $\Theta(\sqrt{n})$. Our algorithm is greedy in nature and uses the intuition provided by the hardness result. In order to better model the prices and values that can arise from practice, we then consider a number of monotonicity conditions. For instance, we say that the values are monotone along the rows if they can be permuted so that v_{ij} increases as i increases. Our results are summarized in Table I, which gives the complexity of multiplicative bidding in different cases against the *individual bidding* optimum.

Unfortunately, the lower bound does not really improve even if the values and prices are monotone for both rows and columns. Nevertheless, we find that the problem becomes tractable given monotone value-over-price ratios (along either row or column). This prompts us to consider a subclass of solutions, *staircases*, that are always feasible.

Building upon this staircase notion, we obtain the more optimistic approximation ratio of $O(\log n)$ assuming that prices are multiplicative, i.e., $p_{ij} = p_i q_j$ and values are monotone along one dimension. These assumptions are justified by empirical data validation (see Section 6.3). At a high level, our algorithm attempts to extract a large subset of the optimum and patch it into a staircase. However, care must be taken to avoid overspending. Indeed, the factor $O(\log n)$ is a compromise between the budget constraint and staircase feasibility.

To apply our algorithms in practice, we must deal with the fact that our monotonicity assumptions hold only in an approximate sense. We address this in Section 6 by providing more robust adaptations of two of our algorithms; these adaptations allow the algorithms to work in a general setting, but still take advantage of the near-monotonicity of the data. We evaluate these algorithms on real search auction data, and show that both have a significant gain over a benchmark inspired by Feldman et al. [2007].

1.3. Related work

This work is most related to the paper of Feldman, Muthukrishnan, Pál, and Stein [2007] in which the authors propose uniform bidding as a means for bid optimization in the presence of budget constraints in sponsored-search ad auctions. There are several differences between this paper and the previous line of work on uniform bidding. Most notably in the multi-dimensional settings, we cannot apply the results of Feldman et al. [2007] and Muthukrishnan et al. [2007]. In fact, as we will observe, our problem in general is inapproximable even for a simple setting.

Previous work on uniform bidding strategies assume that the number of impressions or clicks that the advertiser gets varies as a function of the bid. Here, on the other hand, we assume that bidding results in winning or not winning the impression. While our hardness results directly apply to such more general settings, our approximation algorithm results need an extra step to generalize to this setting. We leave this as an interesting future research direction.

[1]We have also shown that it is $\Omega(n^{1/2-\epsilon})$-hard to approximate against the less stringent *multiplicative bidding* optimum (Lemma 2.4).

As a central issue in online advertising, optimizing under budget constraints has been studied extensively both from publishers' (or search engines') point of view [Mehta et al. 2007; Devanur and Hayes 2009; Goel et al. 2012; Charles et al. 2013; Karande et al. 2013], and from advertisers' point of view [Borgs et al. 2007; Feldman et al. 2007; Rusmevichientong and Williamson 2006; Chakrabarty et al. 2007; Muthukrishnan et al. 2007; EvenDar et al. 2009; Archak et al. 2012]. More closely relevant to this paper, the bid optimization with budget constraints has also been studied from advertisers' perspective: This has been considered either in a repeated auction setting [Borgs et al. 2007], or in the context of broad-match ad auctions [EvenDar et al. 2009], or the case of long-term carryover effects [Archak et al. 2012].

2. HARDNESS RESULTS

We present lower bounds of $\Omega(\sqrt{n})$ and $\Omega(n^{1/2-\epsilon})$ for the multiplicative bidding problem in various natural scenarios. Besides implying that the approximation algorithm in the next section is asymptotically optimal, they also show that the requirements enforced on values and prices in our $O(\log m)$-approximation cannot be easily loosened.

LEMMA 2.1. *There exists an instance such that the gap between multiplicative bidding and individual bidding is $\Omega(\sqrt{n})$, even when the prices are all equal.*

PROOF. Consider the following bad instance where $m = n$:

— prices: $p_{ij} = 1$
— values: $v_{ii} = 1$, $v_{ij} = 0$
— budget: $B = n$

$$(v_{ij}) = \begin{pmatrix} 1 & & 0 \\ & \ddots & \\ 0 & & 1 \end{pmatrix}$$

Hence, we have $\texttt{OPT} = n$ by picking all diagonal cells. We make a crucial observation that realizes the tension to bid multiplicatively. This will be recurrent in this paper:

OBSERVATION 1. *If our bid multipliers capture both (i, i) and (j, j) entries, then at least one of (i, j) and (j, i) is also captured.*

The reason for this is simple. Capturing both (i, i) and (j, j) implies that $r_i \cdot c_i \geq p_{ii} = 1$ and $r_j \cdot c_j \geq p_{jj} = 1$. Thus, we must have $r_i \cdot c_j \geq p_{ij} = 1$ or $r_j \cdot c_i \geq p_{ji} = 1$. In fact, all we need is $p_{ii}p_{jj} = p_{ij}p_{ji}$.

Now we return to the main proof. Suppose that we collect k diagonal entries. Then we must also collect at least $\binom{k}{2}$ off-diagonal entries by the observation. Therefore,

$$B = n \geq \binom{k}{2} \implies \texttt{ALG} = k = O(\sqrt{n}) = O(\texttt{OPT}/\sqrt{n}). \quad \square$$

In light of the last lemma, one might hope that some less pessimistic approximation ratio is attainable under certain conditions. We rule out one such possibility.

Prices and values are *monotone* in both dimensions if the rows and columns can be rearranged so that for any $i \geq i', j \geq j'$, we have $v_{ij} \geq v_{i'j'}$ and $p_{ij} \geq p_{i'j'}$.

LEMMA 2.2. *There exists an instance such that the gap between multiplicative bidding and individual bidding is $\Omega(n^{1/3})$, even when the prices and values are monotone.*

PROOF. The proof is a more elaborate version of Lemma 2.1. Consider the following bad instance where $m = n$:

— $p_{ij} = n^{-1/3}, v_{ij} = 0$ for $i + j < n + 1$ (above antidiagonal)
— $p_{ij} = 1, v_{ij} = 1$ for $i + j = n + 1$ (on antidiagonal)
— $p_{ij} = n^{1/3}, v_{ij} = 1$ for $i + j > n + 1$ (below antidiagonal)
— $B = n$

$$(p_{ij}) = \begin{pmatrix} n^{-1/3} & & 1 \\ & \iddots & \\ 1 & & n^{1/3} \end{pmatrix}, \quad (v_{ij}) = \begin{pmatrix} 0 & & 1 \\ & \iddots & \\ 1 & & 1 \end{pmatrix}$$

Again, we have $\texttt{OPT} = n$ by picking all antidiagonal entries. Our goal is to show that no algorithm achieves a total value of $\omega(n^{2/3})$.

Let k be the number of antidiagonal entries captured. Further let k_1 and k_2 be the numbers of entries captured above and below the antidiagonal respectively. As in Lemma 2.1, we have $k_1 + k_2 \geq \binom{k}{2}$.

On the other hand, the budget constraint dictates that

$$k + k_1 n^{-1/3} + k_2 n^{1/3} \leq n,$$

which gives $k_1 \leq n^{4/3}, k_2 \leq n^{2/3}$ and in turn $k = O(n^{2/3})$. Our claim is immediate since $\texttt{ALG} = k + k_2 = O(n^{2/3})$. $\square$

In the full version, we generalize the construction above to establish an $\Omega(n^{\frac{c-1}{2c-1}})$ gap for an arbitrary positive integer c. Thus the case here corresponds to $c = 2$. By taking $c \to \infty$, we conclude a hardness of $\Omega(n^{1/2-\epsilon})$.

LEMMA 2.3. *There is an instance such that the gap between multiplicative bidding and individual bidding is $\Omega(n^{1/2-\epsilon})$, even when the prices and values are monotone.*

Finally, we show in the full version that the approximation ratio is still dismal even when compared against the less stringent multiplicative bidding optimum.

LEMMA 2.4. *Unless* **NP** $\subseteq$ **BPP***, there is no randomized polynomial-time algorithm that finds a solution of value within a factor of $O(n^{\frac{1-\epsilon}{2}})$ of the optimal multiplicative bidding solution.*

3. TIGHT $O(\sqrt{n})$-APPROXIMATION ALGORITHM FOR THE GENERAL CASE

In this section, we present an $O(\sqrt{n})$-approximation for the multiplicative bidding problem. This matches the lower bound in Lemmas 2.1 and 2.4. Our algorithm will greedily construct at most $O(\sqrt{n})$ disjoint feasible solutions whose union captures all the cells in the individual bidding optimum $\texttt{OPT}$. The approximation promise follows immediately by picking one among these.

We first give an overview of the class of solutions found by the algorithm before describing it formally. Each of the solutions will focus on certain *active* columns. This can be done by bidding 0 on the other columns.

On the other hand, the bid on each active column is just 1, and the bid on row i equals the maximum price p_{ij} among the entries $(i, j) \in \texttt{OPT}$ in an active column j, or simply 0 if there is no such entry.

Observe that this kind of bidding allows us to capture all the $\texttt{OPT}$ entries in the active columns. Our algorithm is stated as Algorithm 1. Notice that the candidate solutions V in the while loop are constructed in a greedy manner.

For the best candidate set V in the algorithm, we bid as follows:

— For $j \notin V$, $c_j = 0$.
— For $j \in V$, $c_j = 1$.
— For $i \in [m]$, $r_i = \max_{(i,j)\in\texttt{OPT}, j\in V} p_{ij}$, or 0 if $\forall j \in V$, $(i, j) \notin \texttt{OPT}$.

As mentioned before, this bidding scheme captures all the $\texttt{OPT}$ cells from columns in V by design. We first show that such a solution is indeed feasible.

ALGORITHM 1: $O(\sqrt{n})$ approximation

```
$U \longleftarrow [n]$;
$candidates \longleftarrow \emptyset$;
while $|U| > 2\sqrt{n}$ do
    Let $V \subseteq U$ be a maximal set of at most $\sqrt{n}$ columns such that
        $$\sum_{i=1}^{m} \max_{(i,j)\in \mathtt{OPT}, j\in V} p_{ij} \le \frac{B}{|V|},$$
    where the $\max$ is 0 if there is no cell from OPT in row $i$ and columns in $V$;
    Add $V$ to $candidates$;
    Remove $V$ from $U$;
end
For each remaining $j \in U$, add $V = \{j\}$ to $candidates$;
Output the best $V$ from $candidates$
```

LEMMA 3.1. *For each candidate V, the total spend is at most B.*

PROOF. If V is chosen in the while loop, then for row i we spend at most $|V|r_i = |V| \max_{(i,j)\in \mathtt{OPT}, j\in V} p_{ij}$. Summing over all rows, we get

$$\sum_{i=1}^{m} |V| \max_{(i,j)\in \mathtt{OPT}, j\in V} p_{ij} \le |V| \cdot \frac{B}{|V|} = B.$$

On the other hand, it is clear that the solution for V chosen after the while loop is a subset of OPT and hence costs at most B. □

Next we establish the approximation guarantee. The intuition behind it is that when V is not large, it consumes a relatively large amount of the budget and hence there cannot be too many $|V| < \sqrt{n}$.

LEMMA 3.2. *If $|V| < \sqrt{n}$ and V is picked before the last candidate in the while loop of the algorithm, we have*

$$\sum_{i=1}^{m} \max_{(i,j)\in \mathit{OPT}, j\in V} p_{ij} \ge \frac{B}{4|V|}. \qquad (*)$$

In particular, the OPT entries from columns in V cost at least $B/4|V|$.

PROOF. By the maximality of V, the fact that V has not reached the size $\sqrt{n}$ implies that no column can be added to V.

Note that inserting a new column j to V increases the L.H.S. of (*) by at most $\sum_{i:(i,j)\in \mathtt{OPT}} p_{ij}$, i.e., the amount OPT spends on j.

Since V is not the last, there must be more than $2\sqrt{n}$ available columns in U, one of which costs no more than $B/2\sqrt{n} \le B/2(|V|+1)$. By the maximality of V, this column cannot be added to V and we must then have

$$\frac{B}{2(|V|+1)} + \sum_{i=1}^{m} \max_{(i,j)\in \mathtt{OPT}, j\in V} p_{ij} > \frac{B}{|V|+1},$$

from which our result follows. □

THEOREM 3.3. *The algorithm above achieves an approximation ratio of $O(\sqrt{n})$.*

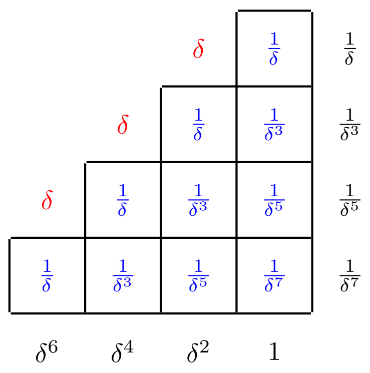


Fig. 1. A *regular* staircase with four rows and four columns. The numbers shown at the bottom and to the right are the column and row multipliers respectively. The numbers shown in blue inside the staircase indicate the effective bids with a lower bound of $1/\delta$. The numbers shown in red outside the staircase are the effective bids with an upper bound of δ. Note that each row or column can have different height or width if we repeat the associated multiplier.

PROOF. It suffices to show that there are $O(\sqrt{n})$ candidates V as they collectively capture all of OPT. Since there can be at most $\sqrt{n}$ of them of size $\sqrt{n}$, and at most $2\sqrt{n}$ candidates V after the while loop, it suffices to bound the number of candidates V with size smaller than $\sqrt{n}$ in the loop.

Let $a_1, \cdots, a_k$ be their sizes. Then trivially $a_1 + \cdots + a_k \leq n$. By Lemma 3.2, their OPT entries cost at least $B/4a_1 + \cdots + B/4a_k \leq B$, hence $1/a_1 + \cdots + 1/a_k \leq 4$. Now by the Cauchy-Schwarz inequality,

$$4n \geq (a_1 + \cdots + a_k)\left(\frac{1}{a_1} + \cdots + \frac{1}{a_k}\right) \geq k^2.$$

This gives $k \leq 2\sqrt{n}$, as desired. $\square$

Finally, we remark that a $O(\sqrt{m})$-approximation can be obtained by swapping the roles of rows and columns. Combining both gives a $O(\min\{\sqrt{m}, \sqrt{n}\})$-approximation.

4. STAIRCASES AS THE BUILDING BLOCK

We introduce an important notion which provides a sufficient condition on the feasibility of a solution. As we shall see, it will be helpful in deriving approximation algorithms for our problem as this subclass of solutions is much easier to work with.

Definition 4.1. A configuration $S \subseteq [m] \times [n]$ is a *staircase* if for any $i, j \in [n]$, its subset of cells in column i is a subset or superset of that in column j.

It is clear that replacing columns by rows in the definition makes no difference. The name staircase originates from the fact that the rows and columns can be permuted in such a way that S indeed resembles a staircase (with possibly uneven step sizes).

LEMMA 4.2. *Ignoring the budget constraint, a staircase S can always be captured exactly.*

PROOF. Figure 1 demonstrates our bidding scheme to capture a simple staircase. Notice that we are bidding $1/\delta^i$ within the staircase and δ^i outside, where i is a positive integer varying across different cells. By making δ sufficiently small, we can capture the staircase exactly.

In general, if S has no entry in any row or column, we set the corresponding bid multiplier to 0 and thus essentially remove it. We then permute the remaining rows and columns such that our staircase takes on a shape as Figure 1. For the rows, we simply bid $1/\delta, 1/\delta^3, \cdots$ successively on each level from top to bottom. In contrast, the bids on the columns are $1, \delta^2, \cdots$ on each level from right to left. $\square$

4.1. 1-approximation for the case of monotone v_{ij}/p_{ij}

The staircase idea immediately implies that our problem can be solved (almost) exactly under one natural assumption.

COROLLARY 4.3. *If the ratios v_{ij}/p_{ij} are monotone in one dimension, then there is a 1-approximation algorithm.*

PROOF. This is immediate since `OPT`, which collects entries with the best v/p ratio as long as the budget B has not been exhausted, will be a staircase when v/p is monotone in one dimension. By Lemma 4.2, `OPT` can be captured exactly. This can be implemented efficiently by selecting the entries with highest v/p ratios one at a time. $\square$

5. $O(\log m)$-APPROXIMATION WHEN PRICES ARE MULTIPLICATIVE AND VALUES ARE MONOTONE

In this section, we make the following two assumptions on the prices and values.

— Multiplicative prices: there are $p_i, q_j > 0$ such that $p_{ij} = p_i \cdot q_j$.
— Monotone values: the values are monotone along one dimension, say, rows. In other words, the rows can be permuted so that $v_{ij} \geq v_{i'j}$ for $i' > i$.

Both of them are necessary in the sense that without either of them, no algorithm can have a good performance as demonstrated by the bad instances in Lemmas 2.1 and 2.3. In the former case, the instance has all prices equal (hence multiplicative) and a gap of $\Omega(\sqrt{n})$, whereas in the latter case, the values are even monotone in both dimensions but the gap is $\Omega(n^{1/2-\epsilon})$. Our assumptions are verified empirically in Section 6.

We now have a nice characterization of the configurations that can be captured.

LEMMA 5.1. *If prices are multiplicative, a configuration $S \subseteq [m] \times [n]$ can be captured if and only if it is a staircase.*

PROOF. One direction simply reiterates Lemma 4.2. For the other direction, let S be a feasible configuration that is not a staircase. Equivalently, there are two columns j_1 and j_2 for which $S \cap ([m] \times \{j_1\})$ is neither a subset nor a superset of $S \cap ([m] \times \{j_2\})$.

Thus, there are some i_1 and i_2 such that $(i_1, j_1), (i_2, j_2) \in S$ and $(i_1, j_2), (i_2, j_1) \notin S$. This is a contradiction since the former implies that $(r_{i_1} c_{j_1})(r_{i_2} c_{j_2}) \geq (p_{i_1} p_{j_1})(p_{i_2} p_{j_2})$ but the latter gives $(r_{i_1} c_{j_2})(r_{i_2} c_{j_1}) < (p_{i_1} p_{j_2})(p_{i_2} p_{j_1})$. $\square$

As a consequence of this lemma, it is sufficient to search for a good staircase within our budget. We present an algorithm fulfilling this objective step by step, each of which, albeit seemingly unrelated, will serve its own purpose. While our derivation establishes an approximation of $O(\log m)$, it can be easily turned into an $O(\log n)$-approximation by swapping the roles of rows and columns and assuming column monotonicity instead of row monotonicity.

5.1. Algorithm

Our algorithm consists of three major steps. An overview is also given in Algorithm 2.

Step 1: Clustering prices. We round *down* all row price multipliers p_i to powers of 2, and permute the rows such that p_i increases from top to bottom. This results in Figure 2(a) where each inner rectangle represents a set of rows with equal p_i. We call these inner rectangles *strips*.

Since p_i is now a power of 2, the prices of two cells in the same column but consecutive strips differs by a factor of at least 2.

Step 2: Finding $\texttt{OPT}(B/4)$. As one may have guessed, our algorithm builds upon the optimal solution to find a good staircase—with one caveat. Instead of basing our solution on the OPT with budget B, we use the OPT with budget $B/4$, denoted by $\texttt{OPT}(B/4)$. The reason for this choice will become apparent in the proof of Lemma 5.3.

We first permute the rows within each strip such that v_{ij} is increasing as we traverse down. This is possible because of value monotonicity. Now $\texttt{OPT}(B/4)$ must consist of width-1 towers that sit on the bottom of our strips, as the prices within the same column in a strip are constant. Figure 2(b) gives one possible $\texttt{OPT}(B/4)$.

In summary, we

— permute the rows in each strip such that v_{ij}'s are increasing from top to bottom; and
— find the cells in $\texttt{OPT}(B/4)$, which must emanate from the bottom of a strip.

Step 3: Constructing a staircase from $\texttt{OPT}(B/4)$. The last step of our staircase construction is made up of two main substeps: we first extract a subset of $\texttt{OPT}(B/4)$ according to a certain parameter $h \in [m]$ and then apply some patching work to transform it into a staircase, denoted $\texttt{ALG}_h$. The final solution ALG will be the best $\texttt{ALG}_h$.

Again, for notational convenience we will use $\texttt{ALG}_h$ and ALG to denote both the set of cells in the solution and their total value.

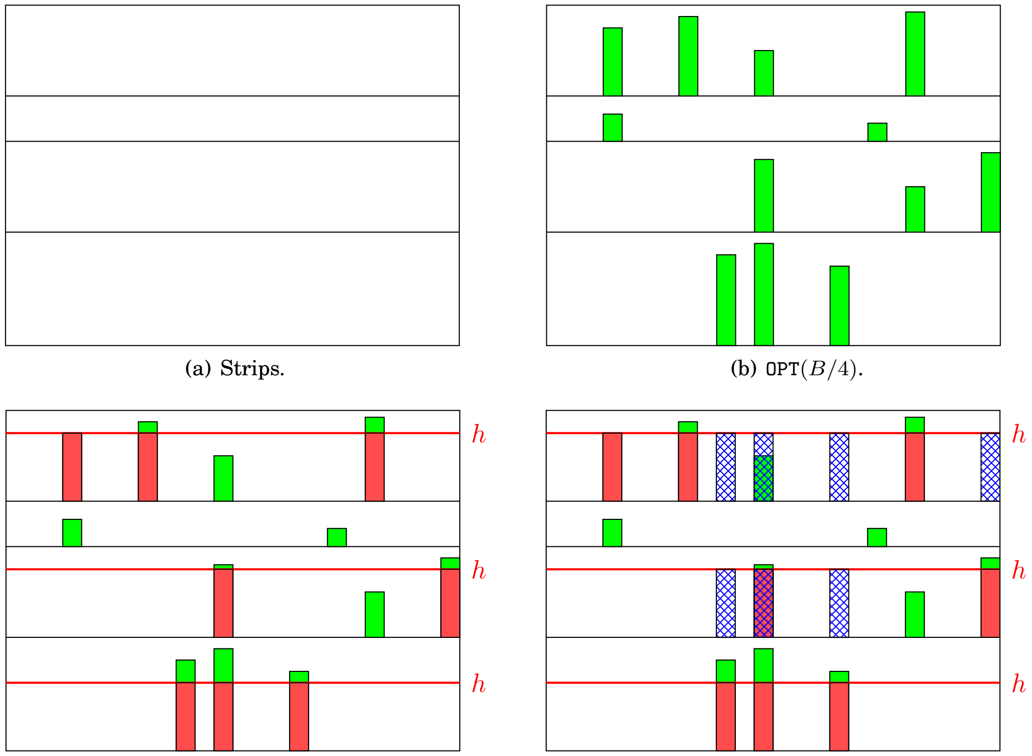
— **Disregard strips of height less than** h
No cells in such strips will be chosen.
— **Take all height-**h **towers**
For the constituent towers of $\texttt{OPT}(B/4)$ with height at least h, we select its $h \times 1$ subtower sitting on the bottom of a strip (Figure 2(c)), and insert them into $\texttt{ALG}_h$.
— **Propagate/copy the height-**h **towers upwards**
If the $h \times 1$ subtower in strip i and column j is chosen in the previous step, we select the corresponding $h \times 1$ subtower in strip i' and column j for all $i' < i$ provided that strip i' has height at least h (Figure 2(d)). They are inserted into $\texttt{ALG}_h$.

Our solution $\texttt{ALG}_h$ thus consists of all the cells selected above. In the figures, these are depicted as red and hatched blue regions. Notice that no cells in strip 2 were selected since its height is smaller than h. A height-h tower can be chosen in both steps.

The algorithm outputs the best $\texttt{ALG}_h$ **for** $1 \leq h \leq m$**, i.e.,** $\texttt{ALG} = \max_{h \in [m]} \texttt{ALG}_h$**.**

ALGORITHM 2: Overview of $O(\log m)$ approximation

```
Step 1:
Round down all of the $p_i$'s to the nearest powers of 2;
Cluster together rows with the same $p_i$;
Reorder the clusters in increasing $p_i$;
Step 2:
Reorder the rows within each cluster in increasing values;
Compute OPT$(B/4)$, the individual bidding optimum with budget $B/4$;
Step 3:
for $h$=1,2,...,$m$ do
    ALG$_h \longleftarrow \emptyset$;
    Insert into ALG$_h$ all height-$h$ towers of OPT$(B/4)$;
    Insert into ALG$_h$ all height-$h$ towers in the strips above the ones in the last line;
end
Output the best ALG$_h$ as ALG
```



(a) Strips. (b) $\texttt{OPT}(B/4)$. (c) Finding towers of height h. (d) Propagating towers upwards.

Fig. 2. The steps in approximating the best multiplicative bidding scheme. The green "towers" in (b) show the cells in $\texttt{OPT}(B/4)$. The red lines in (c) illustrate one candidate height h. Note that the second strip is shorter than h, and therefore does not contribute to $\texttt{ALG}_h$ at all. The red towers in (c) depict the intermediate solution of height h; this is extended to $\texttt{ALG}_h$ in (d) via propagating upwards: see the hatched towers.

5.2. Analysis

LEMMA 5.2. *$\texttt{ALG}_h$ is a staircase.*

PROOF. This is almost by design. For each column, the cells in $\texttt{ALG}_h$ are exactly the height-h towers up to some strip i (excluding those whose height is less than h). This is guaranteed by the propagation operation.

Thus the cells of $\texttt{ALG}_h$ in a column must be a subset or superset of that in another. □

LEMMA 5.3. *$\texttt{ALG}_h$ costs no more than B, i.e., $\sum_{(i,j)\in \texttt{ALG}_h} p_{ij} \leq B$.*

PROOF. First of all, the cells selected in the first substep of step 3 cost at most $B/4$ since they are part of $\texttt{OPT}(B/4)$. We argue that the propagation operation, which is the second substep, spends also at most $B/4$.

Recall that we have rounded the row price multipliers to powers of 2 in step 1. In particular, the prices between two consecutive strips must then differ by a factor of 2 or more. Therefore a height-h tower, when copied upwards, spends at most $1/2^i$ of its cost on the i-th strip above it.

Now summing over all height-h towers from $\texttt{OPT}(B/4)$, the total cost of the propagation operation does not exceed $B/4 \cdot (1/2 + 1/4 + \dots) = B/4$.

Our total cost, which has accounted for all of $\texttt{ALG}_h$, is bounded by $B/4 + B/4 = B/2$. Finally, since we have rounded down prices to powers of 2 in step 1, the actual cost of $\texttt{ALG}_h$ cannot be more than twice of $B/2$. □

LEMMA 5.4. *We have $\texttt{OPT}(B/4) \geq \texttt{OPT} \cdot (1/4 - \epsilon)$.*

PROOF. We will use the small cost assumption $p_{ij}/B < \epsilon$ here.[2]
Recall that $\texttt{OPT}(B')$ collects the cells (i, j) in the order of decreasing v_{ij}/p_{ij} until the budget B' is exhausted. Thus, the extra cells collected in $\texttt{OPT} \setminus \texttt{OPT}(B/4)$, which cost $3B/4$, each have a v_{ij}/p_{ij} ratio no better than those in $\texttt{OPT}(B/4)$.
This implies that $\texttt{OPT} \leq \frac{\texttt{OPT}(B/4)}{1/4-\epsilon}$, since $\texttt{OPT}(B/4)$ has a cost of more than $B \cdot (1/4 - \epsilon)$ by the small cost assumption. □

LEMMA 5.5. *We have $\texttt{ALG}_h \geq \texttt{OPT}(B/4)/2 \log m$ for some h.*

PROOF. Let $\texttt{OPT}_h(B/4)$ be the total values of the $\texttt{OPT}(B/4)$ cells at height h in each strip. We claim that

$$\texttt{ALG}_h \geq h \cdot \texttt{OPT}_h(B/4).$$

This can be seen as follows. Each cell $(i, j) \in \texttt{OPT}_h(B/4)$ is the top of some height-h tower in a strip. Notice that the $h - 1$ cells below (i, j) have values at least v_{ij} by value monotonicity. Our claim then follows from summing over all $(i, j) \in \texttt{OPT}_h(B/4)$.
The rest of the proof is standard. Suppose that no choice of h gives $\texttt{ALG}_h \geq \texttt{OPT}(B/4)/2 \log m$. Then for all h, we have

$$h \cdot \texttt{OPT}_h(B/4) \leq \texttt{ALG}_h < \frac{\texttt{OPT}(B/4)}{2 \log m},$$

which implies

$$\texttt{OPT}(B/4) = \sum_{h=1}^{m} \texttt{OPT}_h(B/4) < \sum_{h=1}^{m} \frac{\texttt{OPT}(B/4)}{h \cdot 2 \log m} \leq \texttt{OPT}(B/4).$$

This is a contradiction. □

Combining all the lemmas, we obtain our main result.

THEOREM 5.6. *Our algorithm gives an $O(\log m)$-approximation.*

PROOF. Lemmas 5.2 and 5.3 establish the feasibility of $\texttt{ALG}_h$. The approximation guarantee follows from Lemmas 5.4 and 5.5. □

We highlight the roles played by different steps of the algorithm. Step 1, which clusters similar price multipliers, ensures that the propagation procedure in Step 3 would result in a convergent geometric sum for the cost. Step 2 computes $\texttt{OPT}(B/4)$ which is essential to achieving our approximation guarantee as well as the budget constraint. Lastly, the propagation procedure in Step 3 also enforces the staircase feasibility.
Our ratio of $O(\log m)$ might not be the most desirable. However, the algorithm likely outperforms its theoretical guarantee in average case. For example, one reason is that the propagation procedure hatches new towers above existing ones but their values are not accounted for in the analysis as they can be negligible in the worst case. In practice, however, such an extreme pattern of values should be rare.

[2]We note that, as mentioned before, our $O(\log m)$-approximation still holds without it.

6. EMPIRICAL STUDY

In this section, we report the results of our experiments. We start by explaining the algorithms implemented, followed by a discussion on their empirical performances. An analysis is also given on the validity of the assumptions made throughout the paper.

6.1. Algorithms and implementation notes

We have tested our algorithms on real world data and compared their performances. We discuss how they are implemented in the rest of this section.

One of the recurrent themes in our paper is monotonicity. In the real world, however, data rarely obey monotonicity perfectly due to presence of noise, among other factors. To apply our algorithms, we must still somehow obtain a decent ordering of the rows (or columns) so that monotonicity approximately holds.

For instance, our monotone value-over-price ratio algorithm from Corollary 4.3 assumes that v_{ij}/p_{ij} increases along the columns (or rows). In essence, each of the n columns induces one possible (partial[3]) permutation of the rows and our job is to aggregate these n candidates into one representative "consensus permutation". To this end, we adopt the procedure in Algorithm 3. This is also used as a subroutine of our $O(\log n)$-approximation implementation to rank values. See section 6.3.2 for a more in-depth discussion of the method.

6.1.1. Staircase algorithm: 1-approximation for Monotone value-over-price ratios. The discussion above sums up how we permute the rows. In the theoretical algorithm, the next step is to simply take OPT, which is a staircase, as the solution. But as mentioned before, this is not always feasible due to imperfect monotonicity. In the implementation, we resort instead to finding a good staircase. More specifically, our solution captures a number of consecutive cells at the bottom of each column.

Thus now we have m possible choices for each column. The resulting optimization problem can be solved by a dynamic program that computes the best solutions formed by the first j columns for different budget thresholds.

6.1.2. Tower building algorithm: $O(\log n)$-approximation for multiplicative prices and monotone values. This is similar to the last implementation. We first cluster together rows with similar prices. For each cluster we then find a permutation of its rows so that values are roughly monotone. At this point we encounter the same issue, namely that $\mathtt{OPT}(B/4)$ is not necessarily a staircase in each price cluster. Moreover, the factor $1/4$ for budget scaling is a rather arbitrary parameter to make the formal analysis go through. Our implementation employs a different view of the algorithm via dynamic programming.

For a given column, our algorithm has the freedom to choose the height-h towers in the first i price clusters, where i ranges from 0 to the number of clusters. Hence there are at most m choices per column. Now the dynamic program simply recursively computes the best solution formed by the first j columns for different budget thresholds.

6.1.3. Uniform bidding. We use an algorithm similar to that proposed by Feldman et al. [2007] as a baseline of how well our algorithms perform. Feldman et al. show that in a large class of bidding problems[4], using a single bid on all keywords guarantees a $1 - 1/e$ approximation, and provides much better performance in practice.

In our adaptation of that algorithm, we find the largest single bid b that can be placed on all cells in the table while respecting the budget constraint. Clearly this can be realized by multiplicative bidding.

[3]The permutations are sometimes partial since the values of a cell can be missing due to data sparsity.

[4]They define their model based on how bids translate to costs and clicks in a collection of second-price auctions by looking at the so-called bid landscapes.

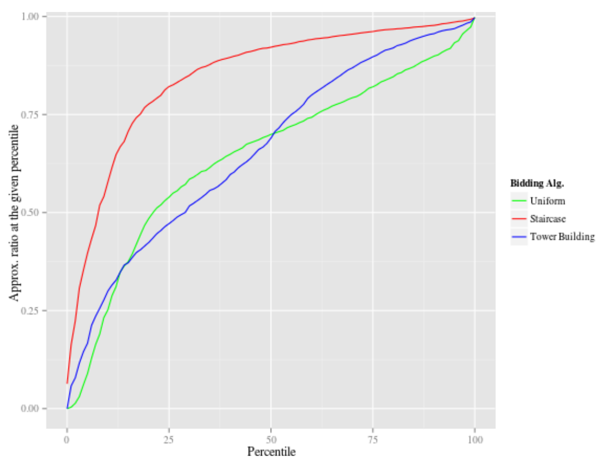


Fig. 3. The performance of the three algorithms. The plot shows what percentage of the individual bidding optimum each algorithm achieves, and quantifies it based on fraction of instances. For instance, at the 50% percentile, in half the instances the performance of the three algorithms is $69\%, 69\%, 92\%$, respectively.

6.2. Results and comparison

We use a dataset of 1000 randomly selected anonymized advertisers using Google AdWords system. Two of the dimensions that these advertisers can provide bid multipliers for are “geo” and “keyword.” The types of impressions that the advertisers are interested in has a wide range as well: from a few up to hundreds of different geo locations, and from tens to thousands of keywords. For our experiments, conversion data acts as a proxy for value, and historic cost-per-click data is used to derive sample price.

We first plot the performance of the three different algorithms, namely uniform bidding, $O(\log n)$ approximation (also called “tower building”) and ratio-based optimization (also called “staircase”). The three curves in Figure 3 show the percentage of the individual bidding optimum that each algorithm can obtain. The uniform bidding approach guarantees 64% of the optimum on average, while this number is 66% and 85% for $O(\log n)$ approximation and ratio-based optimization, respectively. The median performances for the three algorithms are $69\%, 69\%, 92\%$ respectively.

These numbers suggest that the staircase approach is much better than the uniform bidding, whereas the tower building algorithm does not give us any benefits over the uniform bidding. The latter conclusion is not entirely true. Investigating how often the tower building algorithm outperforms the uniform bidding result, we find that the best of the two algorithms provides a mean of 75% and a median of 81% for performance. Figure 4 illustrates this with a histogram of the gain of the two algorithms over the uniform bidding result. Finally we compare the staircase and tower building algorithms. As expected from the above analysis, the former is almost always the better approach. Though the latter outperforms the former in 10% of the instances in our dataset, the gains in these cases are nominal. See Figure 5 for details.

6.3. Validating our assumptions

6.3.1. Are prices multiplicative?. To validate the assumption that prices are multiplicative, we looked at click price data from AdWords, aggregating by the country in which the search query was performed, and the hour of the day. (We did this aggregation over all ad clicks over a week of time in November, 2013.) From this we obtained p_{ij}, the average click price for each combination of country i and hour j. We then looked for two vectors r and c such that the price p_{ij} was well-approximated by $r_i c_j$. To find this vector

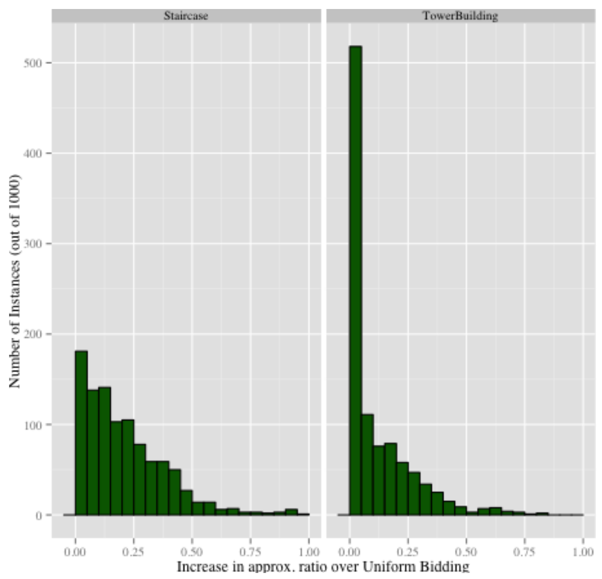


Fig. 4. The gain of our algorithms with respect to uniform bidding. For example, at 0.25 on the x axis, we can read the number of instances where either algorithm could improve uniform bidding by 25%.

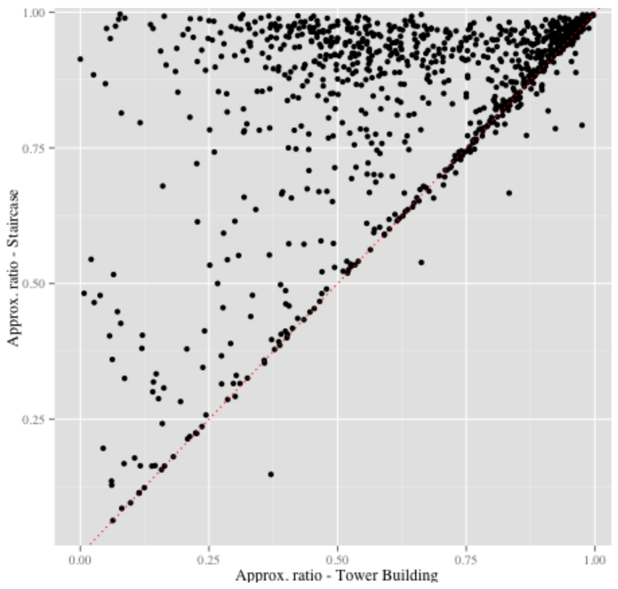


Fig. 5. Comparison between our two algorithms: tower building vs. staircase

we ran a linear regression fitting $\log(p_{ij}) \sim \log(r_1)+\cdots+\log(r_n)+\log(c_1)+\cdots+\log(c_m)$, which results in independent coefficients for each country and each hour of the day. If this is a good fit, it means that $\log(p_{ij}) \approx \log(r_i) + \log(c_j)$, which implies $p_{ij} \approx r_i c_j$.

The regression indeed was a good fit, with virtually every country and hour as significant predictors, and an R^2 value of 0.94. The density plot in Figure 6 shows the actual prices vs. the prices predicted by the regression model, i.e., it plots p_{ij} vs. $r_i c_j$ (note the scale has been changed to $[0, 1]$ for privacy purposes).

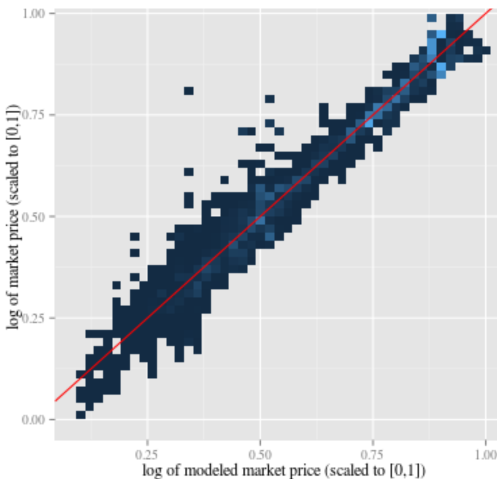


Fig. 6. Verifying that prices are multiplicative. For all i, j, the actual market price p_{ij} (y-axis) is plotted against a predicted market price $r_i c_j$ (x-axis), where r_i and c_j are computed using linear regression. Prices are shown in log scale, under a linear transformation to $[0, 1]$.

6.3.2. Are values and ratios monotone?. For $1 \leq j \leq n$, let $S_j \subseteq [m]$ be the set of entries in column j, and π_j be the permutation of S_j induced by increasing order of the values (or ratios) of S_j. In other words, we have n partial[3] permutations of $[m]$ which we wish to aggregate into one "consensus" permutation π.

Our heuristic in Algorithm 3 was inspired by the algorithm for rank aggregation in Ailon et al. [2008], where the input orderings π_j are complete rather than partial.

In words, the heuristic first constructs a digraph encoding the dominance relationship between every two $i, i' \in [m]$ by a majority vote. Then a random vertex is selected one at a time and used to extend our partial order π. At the end, the remaining incomparable pairs are ordered arbitrarily.

We evaluate how good π is with respect to $\pi_1, \ldots, \pi_n$ by the fraction of agreements:

$$\text{quality}(\pi; \pi_1, \ldots, \pi_n) = \frac{\sum_{j=1}^{n} \#\{(i, i') \in S_j \times S_j | i <_{\pi_j} i', i <_{\pi} i'\}}{\sum_{j=1}^{n} \binom{|S_j|}{2}}.$$

Thus we have quality$(\pi; \pi_1, \ldots, \pi_n) = 1$ if our total order π is perfectly consistent with all of $\pi_1, \ldots, \pi_n$, showing that our values are monotone. On the other hand, for any given *complete* permutation $\pi_1, \ldots, \pi_n$, it is possible to construct a π with quality $\frac{1}{2}$ by choosing a random π_j. This is in fact the well-known folklore 2-approximation for the rank aggregation problem. As an example, feeding our heuristic with random complete permutations will generate a total order with quality $\frac{1}{2}$ in expectation.

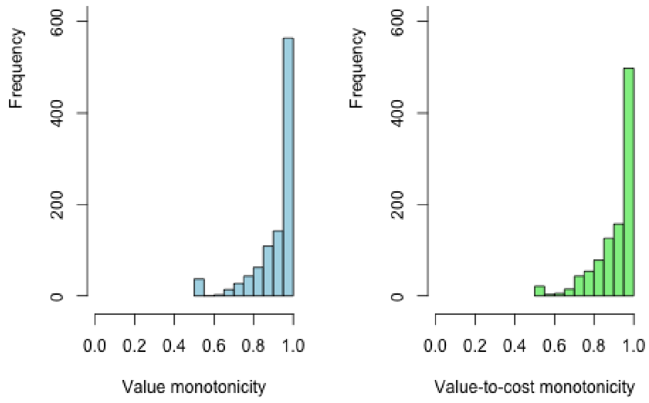
Figure 7 plots a histogram of this quality measure for the heuristic consensus permutations found. Note that this is only a lower bound on how monotone the instance is because perhaps we did not find the best consensus permutation.

7. CONCLUSION AND OPEN PROBLEMS

In this paper we have formulated the multiplicative bidding problem, and characterized its complexity in various cases. In many settings it is $\Omega(n^{1/2-\epsilon})$-hard to approximate. Nonetheless, the problem becomes approximable after imposing appropriate and natural assumptions on the input. A wealth of future work on this new scheme merits study, and we sample a few which we believe are of particular prominence.

ALGORITHM 3: Heuristic for computing consensus permutation

```
Input: partial permutations $\pi_1, \ldots, \pi_n$
Output: total ordering $\pi$
Construct a digraph $D$ over the vertex set $[m]$:
for $i, i' \in [m], i \neq i'$ do
    if no $\pi_j$ contains both $i, i'$ then
        continue;
    end
    if more than half the $\pi_j$'s containing $i, i'$ have $i <_{\pi_j} i'$ then
        insert edge $(i, i')$ to $D$;
    else
        insert edge $(i', i)$ to $D$;
    end
end
Maintain a partial order $\pi$ initialized to be empty:
for $i = 1, 2, \ldots, m$ do
    Let $s_i \in [m]$ be a random vertex of $D$ not processed yet.
    for each parent $p$ of $s_i$ do
        make $p <_\pi s_i$ in $\pi$ if $p, s_i$ are currently incomparable;
    end
    for each child $c$ of $s_i$ do
        make $c >_\pi s_i$ in $\pi$ if $c, s_i$ are currently incomparable;
    end
end
while $\pi$ is not yet a total order do
    find two incomparable $i, i' \in [m]$ and make $i <_\pi i'$ in $\pi$;
end
```



Monotonicity for	25%	50%	Mean	75%
Value	0.8838	0.9697	0.9042	1.0
Ratio	0.8648	0.9500	0.9059	1.0

Fig. 7. Verifying "monotonicity" assumptions in our dataset. The table shows the mean and median as well as the $\frac{1}{4}$ and $\frac{3}{4}$ quantiles for both measurements.

One obvious question is to close the gap for the case of multiplicative prices and monotone values. It is also worthwhile to explore the complexity of the problem under other realistic assumptions (e.g., multiplicative values and correlated values/prices).

Taking a step back, most of our results are concerned with the individual bidding optimum but an advertiser may be more interested in learning how well he *could* be doing rather than understanding the inherent limitation of multiplicative bidding. While we have a hardness of $\Omega(n^{\frac{1-\epsilon}{2}})$ against the multiplicative bidding optimum for the general case, the prospect of better approximations in other cases is not ruled out.

Another interesting direction is to improve our model, or even to propose a new one altogether. One may, for instance, introduce a nonuniform supply constraint. In our model, each cell supplies only one unit of the item. More generally, we can consider the landscape function which specifies the price of an item at different supply levels.

REFERENCES

AILON, N., CHARIKAR, M., AND NEWMAN, A. 2008. Aggregating inconsistent information: ranking and clustering. *Journal of the ACM (JACM) 55,* 5, 23.

ARCHAK, N., MIRROKNI, V., AND MUTHKRISHNAN, S. 2012. Budget optimization of advertising campaigns with carryover effect. In *WINE*.

Bing Ads 2014. Bing ads bid adjustments. http://advertise.bingads.microsoft.com/en-us/help-topic/how-to/moonshot_conc_aboutadvancedbidding.htm/target-customers-with-bid-adjustments.

BORGS, C., CHAYES, J., ETESAMI, O., IMMORLICA, N., JAIN, K., AND MAHDIAN, M. 2007. Dynamics of bid optimization in online advertisement auctions. In *WWW*. 531–540.

CHAKRABARTY, D., ZHOU, Y., AND LUKOSE, R. 2007. Budget constrained bidding in keyword auctions and online knapsack problems. In *SSA*.

CHARLES, D. X., CHAKRABARTY, D., CHICKERING, M., DEVANUR, N. R., AND WANG, L. 2013. Budget smoothing for internet ad auctions: a game theoretic approach. In *ACM Conference on Electronic Commerce*. 163–180.

DEVANUR, N. AND HAYES, T. 2009. The adwords problem: Online keyword matching with budgeted bidders under random permutations. In *EC*.

EVENDAR, E., MANSOUR, Y., MIRROKNI, V., MUTHKIRSHNAN, S., AND NADAV, U. 2009. Bid optimization in BroadMatch ad auctions. In *WWW*.

FELDMAN, J., MUTHUKRISHNAN, S., PÁL, M., AND STEIN, C. 2007. Budget optimization in search-based advertising auctions. In *EC*. ACM, 40–49.

GOEL, G., MIRROKNI, V., AND PAESLEME, R. 2012. Polyhedral clinching auctions and the adwords polytope. In *STOC*.

Google Support 2014. Setting bid adjustments. http://support.google.com/adwords/answer/2732132.

HubSpot 2013. The most important changes to google adwords in 2013. http://blog.hubspot.com/marketing/google-adwords-changes-2013-list.

KARANDE, C., MEHTA, A., AND SRIKANT, R. 2013. Optimizing budget constrained spend in search advertising. In *WSDM*. 697–706.

MEHTA, A., SABERI, A., VAZIRANI, U., AND VAZIRANI, V. 2007. Adwords and generalized online matching. *JACM 54,* 5.

MUTHUKRISHNAN, S., PÁL, M., AND SVITKINA, Z. 2007. Stochastic models for budget optimization in search-based advertising. In *WINE*.

RUSMEVICHIENTONG, P. AND WILLIAMSON, D. 2006. An adaptive algorithm for selecting profitable keywords for search-based advertising services. In *EC*. 260–269.