The Adwords Problem: Online Keyword Matching with Budgeted Bidders under Random Permutations

Nikhil R. Devanur
Microsoft Research
One Microsoft Way
Redmond, WA
nikdev@microsoft.com

Thomas P. Hayes
Dept. of Computer Science
University of New Mexico
Albuquerque, NM
hayes@cs.unm.edu

ABSTRACT

We consider the problem of a search engine trying to assign a sequence of search keywords to a set of competing bidders, each with a daily spending limit. The goal is to maximize the revenue generated by these keyword sales, bearing in mind that, as some bidders may eventually exceed their budget, not all keywords should be sold to the highest bidder. We assume that the sequence of keywords (or equivalently, of bids) is revealed on-line. Our concern will be the competitive ratio for this problem versus the off-line optimum.

We extend the current literature on this problem by considering the setting where the keywords arrive in a random order. In this setting we are able to achieve a competitive ratio of $1 - \epsilon$ under some mild, but necessary, assumptions. In contrast, it is already known that when the keywords arrive in an adversarial order, the best competitive ratio is bounded away from 1. Our algorithm is motivated by PAC learning, and proceeds in two parts: a training phase, and an exploitation phase.

Categories and Subject Descriptors

F.2.m [**ANALYSIS OF ALGORITHMS AND PROBLEM COMPLEXITY**]: Miscellaneous

General Terms

Algorithms

Keywords

Online, Matching, Adwords, Learning, Random Permutation

1. INTRODUCTION

Selling online advertising alongside search results is a multi-billion dollar business [8, 20, 16] and is the major source of revenue for search engines like Google, Yahoo and Microsoft Live Search. Since such systems are (mostly) automated, and handle very large numbers of searches, there is tremendous scope for optimization, which has attracted practitioners and theoreticians alike. Not surprisingly, the theoretical analyses generally consider somewhat simplified abstractions of the actual problem, and ours is no exception. Fortunately, even very simple models have provided interesting insights both from an algorithmic and a game theoretic perspective.

Some of the features that have been considered (or ignored) in the models [20, 8, 19, 16, 18, 4, 7, 17, 1, 9] include: online vs. offline or equilibrium properties, algorithm design vs mechanism design, single keyword vs multiple keywords, incorporating budgets or not, and pay per click vs pay per impression schemes. The problem we consider is an online algorithm design problem, with multiple keywords and budgets, with a pay per impression scheme:

> Suppose there are n bidders with known daily budgets $B_1, B_2, \ldots, B_n$. There are m queries, that come online. (Crucially, the number m is known in advance by the algorithm.) When each query j arrives, bidders $i = 1 \ldots n$ submit bids u_{ij}. The algorithm has to allocate the query to one of the bidders, without knowing the future bids. If query j is allocated to bidder i, then the algorithm collects revenue u_{ij}; however, the total revenue collected from bidder i cannot exceed B_i.
>
> We assume that the *queries* arrive in a random order, i.e., the bids are picked adversarially for all the queries at the start, but at any point, for all queries j, j' not already seen, both the bids $(u_{1j}, u_{2j}, \ldots, u_{nj})$ and $(u_{1j'}, u_{2j'}, \ldots, u_{nj'})$ are equally likely to appear.

Search keywords are categorized as those in the head: ones that appear very often, and those in the tail: ones that appear rarely. In fact, a significant fraction of the queries appear just once. It is this "fat tail" that gives search engines significant advantage over traditional media, which must generally ignore the tail. But it is also the tail that is harder to optimize over. Standard machine learning techniques work well when applied to the head queries, but it is not clear how to use statistical data to learn anything about the tail. Our model is intended to capture exactly the nature of tail queries, since we allow all the bid vectors to be different.

Online algorithms have mostly been analyzed under the *worst case* assumption, namely that the order of the bids could be adversarial as well as the bid amounts. The adwords problem under the worst case assumption was intro-

EC'09, July 6–10, 2009, Stanford, California, USA.

duced by Mehta *et al.* [19], who not only gave a $1 - 1/e$ competitive algorithm[1] but also showed that no randomized algorithm can do better. However, the worst case assumption may be too pessimistic about the kinds of queries a search engine gets.

It is therefore natural to investigate stronger assumptions in order to beat the lower bound that applies in the worst case. There are several models that assume certain randomness in the input, with the goal of capturing the *average case*. Some of these models are

1. Bid vectors are i.i.d from a *known* distribution.
2. Bid vectors are i.i.d from an *unknown* distribution, with the guarantee that the distribution is concentrated on a few vectors.
3. Bid vectors are i.i.d from an *unknown* distribution.

Different models are appropriate for different segments of the keyword market, based on the frequency of the keywords. Models 1 and 2 are more appropriate for the head queries, and are actually quite easy to optimize over. Model 3 is more appropriate for the queries in the tail. The random permutation model is very similar to Model 3, and is cleaner to deal with. In fact, another way to think about the random permutation model is that the bid vectors are sampled *without replacement* from an unknown distribution. Note that in all of Models 1,2 and 3, the optimal offline solution is itself a random variable, whereas in the random permutation model it is not. This allows us to avoid questions such as whether the competitive ratio should be defined as a ratio of expectations or the expectation of a ratio.

[19] asked if an algorithm achieving a competitive ratio of $1-o(1)$ could be designed under a distributional assumption. In this paper, we design such an algorithm under the random permutation model. The take-home message of our result is that the lower bound on the competitive ratio in the worst case depends crucially on the *order* of the bids, not just on the values.

Our focus will be exclusively on the algorithmic problem, ignoring strategic aspects, such as the problem of designing a "truthful" auction mechanism (which also decides the payments in addition to allocating the items). One reason for this is that designing a truthful auction for the *offline* version of our problem is itself a challenging open problem. Sponsored search auctions have other aspects such as multiple slots, second price payments, paying-per-click, and so on. Some of these extensions are straightforward, while extending our results to incorporate other aspects is an interesting avenue for future research.

An implicit practical requirement of such algorithms is that they should be easy to incorporate into the existing systems used by the search engines. Our algorithm should pass this test; indeed existing systems used by many search engines already multiply each bid with a "quality factor." Our algorithm introduces an extra multiplicative factor that accounts for the budget of the bidder.

1.1 Our Results and Techniques

Let OPT denote the best revenue achievable offline. If ALG denotes our algorithm's expected revenue, where the expectation is taken over the random order of the queries, then the competitive ratio of our algorithm is ALG/OPT. Let $b_{\max} = \max_{i,j}\{u_{ij}\}$ be the maximum bid on any query and let $\lambda = \max_{i,i',j}\{u_{i'j}/u_{ij} : u_{ij} \neq 0\}$ be an upper bound on the ratio of the maximum to the minimum non zero bid on any query. Our main result is the following.

THEOREM 1. *There is a $1-\epsilon$ competitive algorithm for all inputs such that*

$$\frac{OPT}{b_{\max}} \geq \Omega\left(\frac{n^2 \log(\lambda/\epsilon)}{\epsilon^3}\right).$$

We first argue that none of our assumptions can be removed if a competitive ratio of $1 - o(1)$ is desired.

- As noted earlier, the lower bound of [19] shows that, without our random permutation assumption, one cannot get a competitive ratio better than $1 - 1/e$.
- The algorithm needs to know the (approximate) number of queries in advance. In particular, the following example shows that if m is not known in advance, then the competitive ratio is bounded away from 1 even when there are only 2 bidders and 2 keywords, a and b, each of which occurs $m/2$ times. The bids are $(1, 0)$ for a and $(2, 1)$ for b. Each bidder has a budget of 150. Now if $m = 100$, then OPT gives all 50 b's to bidder 1 (in addition to all the a's). However, if $m = 200$ then OPT only gives 25 b's to bidder 1 and 75 b's to bidder 2. After seeing 100 keywords, any allocation has to significantly deviate from OPT (and have a significantly lower revenue) in at least one of the two values of m. Thus, an online algorithm that does not know m has to have a competitive ratio bounded away from 1.
- Finally, an assumption such as OPT $\geq f(n, \epsilon, b_{\max})$ is necessary, as was shown by Goel and Mehta [10]. The exact dependence required on n and ϵ is an open question.

Thus, our result is optimal in the weak sense that none of the assumptions of our model can be dropped. Also, note that our assumption only states that the optimum is sufficiently large relative to the maximum bid, and thus allows some (but not all) of the bidders to have a bid-to-budget ratio approaching 1.

Our algorithm (and the analysis) is very much like a PAC-learning algorithm, although PAC-learning is typically used for classification and not optimization. The algorithm uses the bids from the first few queries as a training set to learn how to allocate the other queries. However, since the bids could all be distinct, the earlier bids do not quite tell you anything about the later bids. In other words, you cannot learn the distribution of bids. The key insight used in the algorithm is that one only needs to learn weights for the bidders such that multiplying their bids by their weights and assigning the queries to the highest weighted bidder is an almost optimal assignment. The algorithm uses the first ϵ fraction of the queries to learn these weights, and uses them on the rest of the queries.

One can draw an analogy with PAC-learning for the analysis as well: typically the sample complexity of a PAC learning algorithm depends on the size of the hypothesis class: the "smaller" the hypothesis class, the better. Given the appropriate sizes, a hypothesis that is good on the sample is also good on the entire distribution. Here the class of algorithms

[1] assuming that the budgets are much larger than the bids.

that we use is parameterized by the weights, and the set of all possible weights plays the role of the hypothesis class. The analysis involves showing that a set of weights that is optimal on the sample is, with high probability, close to optimal on the entire set of queries. We apply standard techniques from PAC learning; in particular, concentration bounds and covering arguments. However, the objective function in our setting (the profit of the mechanism) is rather badly behaved, which will present obstacles to our approach. We overcome these via the following techniques:

- We use LP duality and complementary slackness conditions to reduce the problem of estimating the profit function to that of estimating n *simpler* functions, to which standard concentration inequalities can be applied.
- We use non-uniform error rates across these functions that are carefully picked and the appropriate concentration inequalities to get the eventual result.
- We use "smoothing" techniques to replace the original profit function with something more tractable. In Section 4, we present two different approaches to this smoothing, one based on random perturbations, and one based on techniques from convex programming. These methods allow us to break ties appropriately (when two bidders have the same weighted bid), and also are important for our covering arguments.

Previous approaches to the problem also critically use LP duality, but have been entirely combinatorial in nature. It is likely that the approach and techniques introduced here are applicable to other problems as well.

1.2 Other Related Work

Buchbinder, Jain and Naor [5] also gave a $1-1/e$ competitive algorithm for the adwords problem using a primal-dual framework. They use the duals in order to determine the primal allocation, which is similar to our algorithm. However, they update the duals after each query, while we calculate the duals just once after an initial sampling. Also, the techniques used in the two papers for the analysis of the competitive ratio are entirely different.

Goel and Mehta [10] considered the random permutation model, and showed that the greedy algorithm achieves a competitive ratio of $1-1/e$, which is tight. They also showed that for some small instances, no deterministic algorithm can achieve a ratio of better than 3/4, while no randomized algorithm can do better than 5/6. Further, they showed that the algorithm of [19] has a competitive ratio $< .81$.

The history of online matching problems goes back to Karp, Vazirani and Vazirani [13] which was one of the very first instances of competitive analysis. They considered the online bipartite matching problem, where the bids are all 0 or 1, and the budgets are all 1, and they gave a $1-1/e$ competitive [2] algorithm for it. Kalyanasundaram and Pruhs [12] gave a $1-1/e$ competitive algorithm for the online b-matching problem (as $b \to \infty$), where the bids are still 0 or 1, but the budgets are all b.

Random permutation models have also been considered in other problems, perhaps most popular in the classic secretary problem and its generalizations [6, 2, 14]. Apart from the similarity about random ordering of the inputs, the two problems seem to be different.

There seem to be similarities between our paper and a paper by Balcan et. al. [3] (which solves a mechanism design problem), especially in the form of the results, and some of the techniques. While these high-level similarities are not surprising–since [3] is also inspired by learning theory– a closer look reveals no deeper connection. In a nutshell, their paper considers a very general class of problems and is therefore independent of any structure in the problem, while the issues that arise and the techniques used in solving the problem in this paper are specific to the nature of the problem.

[2] A gap in their analysis was discovered by [15], and an alternate proof was given in [10].

2. ALGORITHM

The goal of the algorithm is to use the earlier bids to gain some information about how to assign the later bids. The algorithm learns weights α_i for each bidder, such that assigning query j to $\arg\max_i\{u_{ij}(1-\alpha_i)\}$ is close to optimal. The existence of such weights follows from LP duality; we outline the existence proof below. The offline problem can be formulated as the integer program

$$\begin{aligned} \max \quad & \sum_{i,j} u_{ij}x_{ij} \\ \text{s.t. for all } i, & \sum_j u_{ij}x_{ij} \le B_i \\ \text{and for all } j, & \sum_i x_{ij} \le 1. \\ & x_{ij} \in \{0,1\}. \end{aligned}$$

Consider the LP relaxation of the above integer program, that allows the x_{ij}s to take on fractional values in $[0,1]$. Since $\mathrm{OPT}/b_{\max}$ is large, the fractional and integral optima are close to each other. Henceforth we will work with the relaxed LP, and ignore the rounding issues.

The dual of the above LP is

$$\begin{aligned} \min \quad & \sum_i \alpha_i B_i + \sum_j p_j \\ \text{s.t. } \forall\, i,j, \quad & p_j \ge u_{ij}(1-\alpha_i). \end{aligned}$$

Note that, at the optimum, $\forall\ j,\quad p_j = \max_i u_{ij}(1-\alpha_i)$. Thus, one can think of the dual objective as just a function of the α_i's. Thus we define

$$D(\alpha) = \sum_i \alpha_i B_i + \sum_j \max_i u_{ij}(1-\alpha_i).$$

By complementary slackness, if (α, p) minimizes the dual LP, and x is the optimal allocation to the primal LP, then

$$x_{ij} > 0 \text{ implies } p_j = \max_i u_{ij}(1-\alpha_i),$$

and hence, given the optimal α, we should allocate item j to bidder $\arg\max_i u_{ij}(1-\alpha_i)$. Thus, if we use the weights $(1-\alpha_i)$ as multiplicative discount factors on the bids, we obtain the optimal allocation, as claimed earlier.

Our algorithm uses the first ϵm queries to try to guess these weights, in a way reminiscent of PAC-learning algorithms: it selects the α that minimizes the restriction of D to the observed bids. More precisely, for all subsets S, of size ϵm, define

$$D(\alpha, S) := \sum_i \alpha_i \epsilon B_i + \sum_{j\in S} \max_i u_{ij}(1-\alpha_i).$$

We now present our core algorithm, Learn-Weights.

Algorithm 2.1: LEARN-WEIGHTS(ϵ)

for $j \leftarrow 1$ **to** ϵm
  Observe the bids u_{ij}.
  Allocate item j arbitrarily (e.g. all $x_{ij} = 0$).
Let $\alpha^* := \arg\min_\alpha\{D(\alpha, S)\}$.
for $j \leftarrow \epsilon m + 1$ **to** m
  Observe the bids u_{ij}.
  Give item j to the bidder maximizing $u_{ij}(1 - \alpha_i^*)$.

It can happen that there are ties; that is, $\arg\max_i\{u_{ij}(1-\alpha_i^*)\}$ may not be uniquely defined. For now, we will ignore this issue, and pretend that such ties never occur (this would be the case if, for instance, the bid vectors u_j were in general position). We will discuss ways to remove this assumption in Section 4, resulting in two improved variants of the above algorithm.

3. COMPETITIVE ANALYSIS

Recall that the main goal for us is to prove that $\alpha^* = \arg\min_\alpha\{D(\alpha, S)\}$ gets enough revenue on the rest of the queries. The main steps in showing this are:

- Find sufficient conditions on α under which the profit of the algorithm using α is guaranteed to be close to OPT (Lemma 2).
- Show that α^* satisfies these conditions (Lemma 5).

We now motivate why the sufficient conditions are as required in Lemma 2. We first define indicator variables that specify the allocation rule that is used in the algorithm; let $x_{ij}(\alpha) = 1$ if $i = \arg\max_i\{u_{ij}(1 - \alpha_i)\}$, and 0 otherwise. Now let the profit obtained by using α on the entire set of queries be $P(\alpha) = \sum_i \min\{B_i, \sum_j u_{ij}x_{ij}(\alpha)\}$. The function $P(\alpha)$ is not well-behaved: it is a step function whose level sets are projections of the facets of the feasible polytope for the dual polytope. The values may oscillate up and down many times along a straight line. On the projections of the lower-dimensional facets, there exist items for which there is not a unique maximum discounted bid $u_{ij}(1 - \alpha_i)$, and hence we have not uniquely specified an allocation $x_{ij}(\alpha)$. In general, it is possible to find allocations on these facets for which the profit $P(\alpha)$ exceeds the values on the adjoining full-dimensional facets.

For the above reasons we cannot directly argue about P. Instead, we define the following functions that are easier to argue about: let $R_i(\alpha) := \sum_j u_{ij}x_{ij}(\alpha)$, be the revenue obtained by bidder i on using the allocation give by α, if there were no budget constraints. The sufficient conditions that we mentioned amount to the fact that the functions R_i are estimated well enough by the sample. Two things are of importance here:

- $R_i(\alpha)$ is a sum of numbers, one for each query. This form of R_i lets us apply concentration inequalities to bound the probability that R_i is estimated well by its restriction to the sample.
- We allow the error in estimating R_i to be different for different bidders, as long as the total error is relatively small. Insisting on uniform error bounds gives a worse overall result.

Before we formalize the above, we need to introduce some more notations. The general rule is that a function with subscript i refers to those terms that correspond to bidder i, and an argument of S defines the restriction of the function to the queries in S. This includes scaling the budgets appropriately, wherever relevant. Thus, we define

$$\begin{aligned}
R_i(\alpha, S) &:= \sum_{j \in S} u_{ij}x_{ij}(\alpha).\\
R(\alpha) &= \sum_i R_i(\alpha)\\
P_i(\alpha) &:= \min\{B_i, \sum_j u_{ij}x_{ij}(\alpha)\} = \min\{B_i, R_i(\alpha)\},\\
P_i(\alpha, S) &:= \min\{|S|B_i/m, R_i(\alpha, S)\},\\
P(\alpha, S) &:= \sum_i P_i(\alpha, S),\\
D_i(\alpha) &:= \alpha_i B_i + (1 - \alpha_i)R_i(\alpha)\\
D_i(\alpha, S) &:= \alpha_i|S|B_i/m + (1 - \alpha_i)R_i(\alpha, S).
\end{aligned}$$

With the above definitions, $D(\alpha) = \sum_i D_i(\alpha)$ and $D(\alpha, S) = \sum_i D_i(\alpha, S)$. Let S^c denote the complement of our sampled set, namely, $\{1, \ldots, m\}\backslash S$. We now formally prove the claim above in the following lemma.

LEMMA 2. *If for all i, $|R_i(\alpha^*, S) - \epsilon R_i(\alpha^*)| \le t_i$, and $\sum_i t_i \le \epsilon^2 \max\{OPT, R(\alpha^*)\}$ then $P(\alpha^*, S^c) \ge (1-O(\epsilon))OPT$.*

Proof: The main idea is that the hypothesis of the lemma translates to the fact that α^* satisfies the complementary slackness conditions for the entire set, approximately. This guarantees that $P(\alpha^*)$ is close to OPT. In the proof, for simplicity of notation, we drop α^* from the arguments, since that is the only α we consider. Also let $a_i = t_i/\epsilon$.

We first show that $P \ge (1 - O(\epsilon))$OPT. A sufficient condition for this is that

$$\max\{R, D\} - P \le \sum_i a_i. \tag{1}$$

Let us see why. Note that $P \le \text{OPT} \le D$, by weak duality. Therefore replacing $\max\{R, D\}$ by R and P by OPT in (1) and using the hypothesis of the lemma implies that $R - \text{OPT} \le \epsilon R$, and so $R \le \text{OPT}/(1-\epsilon)$. Thus $\max\{\text{OPT}, R\} \le \text{OPT}/(1 - \epsilon)$. (1) can now be used once again to conclude that $\text{OPT} - P \le \epsilon\text{OPT}/(1 - \epsilon)$. This establishes sufficiency of (1).

We now prove that for all i, $\max\{R_i, D_i\} - P_i \le a_i$ which implies (1). Recall that $D_i = \alpha_i^* B_i + (1 - \alpha_i^*)R_i$, and $P_i = \min\{B_i, R_i\}$. We consider two cases:

Case 1 $\alpha_i^* > 0$.

$\max\{D_i, R_i\} - P_i \le \max\{B_i, R_i\} - \min\{B_i, R_i\} = |B_i - R_i|$. Since $\alpha_i^* > 0$, from complementary slackness conditions on the LP restricted to queries in S, we have that $R_i(S) = \epsilon B_i$. Now from the hypothesis in the lemma we have that $|\epsilon B_i - \epsilon R_i| \le \epsilon a_i$, which is the same as $|B_i - R_i| \le a_i$.

Case 2 $\alpha_i^* = 0$. $D_i = R_i \le R_i(S)/\epsilon + a_i$. $R_i(S) \le \epsilon B_i$, so $D_i \le B_i + a_i$. $D_i = R_i$ and $D_i \le B_i + a_i$ implies $D_i - P_i \le a_i$.

We now prove that $P(S^c) \ge (1 - \epsilon)P - \sum_i t_i$. The hypothesis of the lemma says that $R_i(S^c) > (1 - \epsilon)R_i - t_i$ and

so

$$\begin{aligned} P_i(S^c) &= \min\{(1-\epsilon)B_i, R_i(S^c)\} \\ &\geq \min\{(1-\epsilon)B_i, (1-\epsilon)R_i - t_i\} \\ &\geq (1-\epsilon)P_i - t_i. \end{aligned}$$

Since $\sum_i t_i \leq \epsilon^2 \max\{\mathrm{OPT}, R\}$, this only contributes second order error terms. □

We will use the following concentration inequality on the sum of a random subset of a set of real numbers, which is an easy variant of Bernstein's inequality.

LEMMA 3. *Let $Y = (Y_1, \ldots, Y_m)$ be a vector of real numbers, and let $0 < \epsilon < 1$. Let S be a random subset of $[m]$ of size ϵm, and set $Y_S := \sum_{j \in S} Y_j$. Then, for every $0 < \delta < 1$,*

$$\mathbf{Pr}\left(|Y_S - \mathbf{E}Y_S| \geq \frac{2}{3}\|Y\|_\infty \ln\left(\frac{2}{\delta}\right) + \|Y\|_2\sqrt{2\epsilon \ln\left(\frac{2}{\delta}\right)}\right) \leq \delta.$$

Proof: Let $X_1, X_2, \ldots, X_{\epsilon m}$ be ϵm i.i.d. random variables each of which is equal to one of the Y_j's uniformly at random. Let $X = \sum_{i=1}^{\epsilon m} X_i$. Note that $\epsilon\|Y\|_2^2$ is an upper bound on the variance of X, and $|X_i| \leq \max_j |Y_j| = \|Y\|_\infty$. X and Y_S are both sums of ϵm random values that are sampled with and without replacement respectively from the same set. Thus, Theorems 3 and 4 in [11] imply that for all $t > 0$,

$$\mathbf{Pr}(|Y_S - \mathbf{E}Y_S| \geq t) \leq 2\exp\left(\frac{-t^2}{2(\epsilon\|Y\|_2^2 + (\|Y\|_\infty t/3))}\right).$$

The desired value for t can be found by setting the right-hand side equal to δ and applying the quadratic formula. The slightly simpler expression in the conclusion of the Lemma follows from the inequality $\sqrt{a^2+b} \leq a + \sqrt{b}$, valid for all $a, b \geq 0$. □

We want to use the above lemma to argue that all the $R_i(\alpha^*, S)$'s are concentrated around their expected value as required by Lemma 2. However, since α^* is itself a function of the sample S, we cannot directly apply the lemma. Instead we apply the lemma to all possible values of α^* and the take a union bound on the failure probability. But then, there are infinitely many possible values of α^*, so we cannot quite do this. Instead, we apply a union bound on a sufficiently dense, but finite subset of α's and argue that this is sufficient. For this purpose we define the following, analogous to the notion of ϵ-nets in real analysis.

DEFINITION 4. *$\Lambda \subseteq [0,1]^n$ is an (x, ϵ)-net for an allocation rule $x_{ij}()$ and an $\epsilon > 0$, if for all $\alpha \in [0,1]^n$, there exists an $\alpha' \in \Omega$ such that $|x_{ij}(\alpha) - x_{ij}(\alpha')| \leq \epsilon$ for all i, j.*

We now prove our main lemma which guarantees that α^* satisfies the hypothesis in Lemma 2, and which in turn implies that the revenue of the algorithm is close to the optimum, given an appropriate ϵ-net. We defer the construction of the ϵ-net to Section 4.

LEMMA 5. *If Λ is an (x, ϵ)-net and*

$$\frac{OPT}{b_{\max}} \geq \Omega\left(\frac{n\log(n|\Lambda|/\epsilon)}{\epsilon^3}\right)$$

then $ALG \geq (1-\epsilon)OPT$.

Proof: For each $1 \leq i \leq n$ and $\alpha \in \Lambda$, we define corresponding "bad" events, $\mathcal{B}_{i,\alpha}$, which are of the form

$$|R_i(\alpha, S) - \epsilon R_i(\alpha)| > t_{i,\alpha}$$

where $t_{i,\alpha}$ is such that $\mathbf{Pr}(\mathcal{B}_{i,\alpha}) \leq \delta = \epsilon/n|\Lambda|$. The appropriate value of $t_{i,\alpha}$ is given by Lemma 3.

$$t_{i,\alpha} = \frac{2}{3}b_{\max}\ln\left(\frac{2}{\delta}\right) + \|R_i(\alpha)\|_2\sqrt{2\epsilon\ln\left(\frac{2}{\delta}\right)},$$

where by abuse of notation we let $\|R_i(\alpha)\|_2$ be the l_2 norm of $(u_{ij}x_{ij}(\alpha))_j$.

We first show that this choice of $t_{i,\alpha}$ satisfies the hypothesis in Lemma 2. When you sum the $t_{i,\alpha}$'s, the first terms in the above expression contribute $O\left(nb_{\max}\ln(\frac{1}{\delta})\right)$ which is less than $\epsilon^3 OPT$ given $\frac{\mathrm{OPT}}{b_{\max}} \geq \Omega\left(\frac{n\log(1/\delta)}{\epsilon^3}\right)$. In order to bound the contribution of the second terms, we use the following two inequalities:

$$\|R_i(\alpha)\|_2 \leq \sqrt{b_{\max}R_i(\alpha)} \quad \text{and}$$

$$\sum_i \sqrt{R_i(\alpha)} \leq \sqrt{n\sum_i R_i(\alpha)} = \sqrt{nR(\alpha)}.$$

The second inequality is obtained by applying the Cauchy-Schwarz inequality to the vector of all ones and $\left(\sqrt{R_i(\alpha)}\right)_i$. Now combining these,

$$\sum_i \|R_i(\alpha)\|_2\sqrt{2\epsilon\ln\left(\frac{2}{\delta}\right)} \leq \sqrt{O(nb_{\max}\epsilon\log(1/\delta)R(\alpha))}.$$

Once again using the fact that $\frac{\mathrm{OPT}}{b_{\max}} \geq \Omega\left(\frac{n\log(1/\delta)}{\epsilon^3}\right)$, we obtain

$$\sum_i t_{i,\alpha} \leq O(\epsilon^2)\max\{\mathrm{OPT}, R(\alpha)\},$$

which is as per the hypothesis of Lemma 2.

If $\alpha^* \in \Lambda$, then by simply applying a union bound over all $\alpha \in \Lambda$ and i, we get that with probablity $\geq 1-\epsilon$, none of the events $\mathcal{B}_{i,\alpha}$ happen. Thus we can apply Lemma 2, to conclude that $P(X(\alpha^*), S^c) \geq (1 - O(\epsilon))\mathrm{OPT}(S^c)$.

Suppose $\alpha^* \notin \Lambda$. However, because Λ is an (x,ϵ)-net, there exists $\widehat{\alpha} \in \Lambda$ such that, for all i, j, $|x_{ij}(\widehat{\alpha}) - x_{ij}(\alpha^*)| \leq \epsilon$. Now, assuming the bad event $\mathcal{B}_{i,\widehat{\alpha}}$ does not occur, we have, by the triangle inequality,

$$\begin{aligned} |R_i(\alpha^*, S) - \epsilon R_i(\alpha^*)| &\leq |R_i(\widehat{\alpha}, S) - \epsilon R_i(\widehat{\alpha})| \\ &+ |R_i(\widehat{\alpha}, S) - R_i(\alpha^*, S)| \\ &+ \epsilon|R_i(\widehat{\alpha}) - R_i(\alpha^*)| \\ &\leq t_{i,\widehat{\alpha}} + \epsilon R_i(\alpha^*, S) + \epsilon^2 R_i(\alpha^*) \end{aligned}$$

and this is,

$$\leq t_{i,\widehat{\alpha}} + O\left(\epsilon^2 R_i(\alpha^*)\right).$$

Summing over i, we find that we can again apply Lemma 2 to obtain the desired result.

□

4. AVOIDING TIES

In the description of our algorithm, we assumed that there are never any ties, where $\arg\max_i u_{ij}(1-\alpha_i^*)$ is not uniquely

defined. In fact, it is important to our arguments that such ties not be allowed to occur. However, in practice, such ties might occur frequently, for various possible reasons. We present two approaches to resolve this issue. One is based on introducing random perturbations, and the other involves a *smoothing* technique from convex programming.

4.1 Random perturbations

Our first approach to resolving this problem is based on the observation that, if the bid vectors u_j are in general position in $\mathbf{R}^n$, then for any α, there can be at most $n-1$ ties. Since $nb_{\max} < \epsilon \mathrm{OPT}$, it doesn't really matter how we break these ties. Unfortunately, we cannot assume the bid vectors are in general position.

To get around this, suppose we choose, in advance, a tiny perturbation $\xi_{i,j}$ to be added to each bid u_{ij}. These will be chosen independently and uniformly at random from the interval $[0, \zeta]$, where $\zeta = O(\epsilon/m)$. Because the perturbations are chosen independently from continuous distributions, the perturbed bid vectors will be in general position with probability one. Because the perturbations are small, the exact amounts of the perturbations will have a negligible effect on the profit for any α. Indeed, if so desired, this effect can be made exactly zero by instead defining ζ as an infinitessimal.

Effectively, the perturbations, by breaking ties, may split some degenerate facets of the feasible polytope for the dual LP into possibly very many small non-degenerate facets. Whereas such an original degenerate facet may have a much larger profit achievable than its neighboring facets, after the perturbations, two adjacent facets can differ in their assignment for at most n items, which means the profit function makes small jumps across neighboring facets.

Note that, since the perturbations are independently sampled, it makes no difference whether we think of these perturbations as being chosen before or after the query order is randomized.

The only remaining question is how these random perturbations affect our earlier lemmas regarding the concentration of $P(\alpha^*)$. Here is one answer.

THEOREM 6. *Suppose the bids are such that*

$$\frac{OPT}{b_{\max}} \geq \Omega\left(\frac{n^2 \log(nm)}{\epsilon^3}\right).$$

Then, with probability 1*, the random perturbations are such that our basic algorithm is* $1-\epsilon$ *competitive on the perturbed input. Moreover, as long as the perturbations are multiplicative and chosen continuously from a range* $[1-O(\epsilon), 1+O(\epsilon)]$*, this implies our modified "with perturbations" algorithm is* $1-O(\epsilon)$ *competitive on the original input.*

Proof:

Note that the optimum α^* for $D(\alpha, S)$ always occurs at the projection of some vertex of the feasible polytope for $D(\alpha)$. Since this polytope is defined by nm linear inequalities, which project down to linear inequalities in n dimensions, the number of vertices they define is at most $\binom{nm}{n} \leq (nm)^n$. Since this set is actually guaranteed to contain α^*, it is an $(x, 0)$-net, and in particular an (x, ϵ)-net. The desired result now follows by Lemma 5. □

Unfortunately, the above bound depends on the number of keywords, m. It is not obvious to us whether this dependence is a necessary part of the above perturbation approach. In the next section, we will present a different approach which does avoid a dependence on the number of keywords.

Algorithm 4.1: LEARN-WEIGHTS-PERTURBED(ϵ, η)

for $j \leftarrow 1$ **to** ϵm
  Observe the bids u_{ij}.
  For each i, sample $\xi_{i,j}$ randomly from $[0, \eta]$.
  Record $\widetilde{u}_{ij} = u_{ij}(1 - \xi_{i,j})$.
  Allocate item j arbitrarily (e.g. all $x_{ij} = 0$).
Let α^* be the $\alpha \geq 0$ minimizing
  $\sum_{i=1}^{n} \alpha_i B_i + \sum_{j=1}^{\epsilon m} \max_i (1 - \alpha_i)\widetilde{u}_{ij}$
for $j \leftarrow \epsilon m + 1$ **to** m
  Observe the bids u_{ij}.
  For each i, sample $\xi_{i,j}$ randomly from $[0, \eta]$.
  Let $\widetilde{u}_{ij} = u_{ij}(1 - \xi_{i,j})$.
  Give item j to the bidder i maximizing $\widetilde{u}_{ij}(1 - \alpha_i^*)$.

4.2 Smoothing

Another approach to breaking ties is to use a smoothed version of the dual LP to determine α^*, and the allocation rule. Actually, $u_{ij}x_{ij}(\alpha)$ can be thought of as a sub-gradient of the function $\max\{u_{ij}(1-\alpha_i)\}$, and when there are ties, the sub-gradient is not unique, because the function is not differentiable at that point. The smoothed version of this is guaranteed to be convex and differentiable. By going to a smoothed version, we make sure that there there is a uniquely determined gradient at all points. This gradient now determines the allocation. We also need to make sure that this smoothing has some additional properties: it satisfies the complementary slackness conditions, it does not introduce substantial error, and the rate of change of the allocation itself is bounded appropriately.

Consider the following convex program, where for all j, C_j is some large constant. This is a smoothed version of the dual LP. We let the constraints $p_j \geq u_{ij}(1-\alpha_i)$ be violated, but add a penalty for the violation.

$$\begin{aligned} \min \quad & \sum_i \alpha_i B_i + \sum_j p_j + \sum_{i,j} \frac{C_j}{2}\theta_{ij}^2 \\ \text{s.t. } \forall\, i,j, \quad & p_j \geq u_{ij}(1 - \alpha_i - \theta_{ij}), \\ & \alpha_i, p_j \geq 0. \end{aligned}$$

We obtain the dual of this convex program (which will be the smoothed version of the primal LP), using Lagrangian duality. Define the Lagrangian function, with multipliers x_{ij},

$$\begin{aligned} \tilde{L}(\alpha, \boldsymbol{p}, \Theta, X) = \quad & \sum_i \alpha_i B_i + \sum_j p_j + \sum_{i,j} \frac{C_j}{2}\theta_{ij}^2 \\ & + \sum_{i,j} x_{ij}\left(u_{ij}(1 - \alpha_i - \theta_{ij}) - p_j\right). \end{aligned}$$

Now define the smoothed version of the primal LP by considering $\min_{\alpha, \boldsymbol{p}, \Theta} \tilde{L}$. Consider the terms involving p_j, that is, $p_j(1 - \sum_i x_{ij})$. If the second term is negative, then the minimum is $-\infty$ since we can make p_j go to ∞. Otherwise the minimum is 0. Similarly, consider the terms involving α_i, that is, $\alpha_i(B_i - \sum_j u_{ij}x_{ij})$. The minimum is either $-\infty$

or zero depending on whether the second term is negative or positive. Finally, consider the terms involving θ_{ij}, that is, $\frac{C_j}{2}\theta_{ij}^2 - u_{ij}x_{ij}\theta_{ij}$. This is minimized at $\theta_{ij} = u_{ij}x_{ij}/C_j$, and the minimum value is $-\frac{(u_{ij}x_{ij})^2}{2C_j}$. Therefore the primal convex program is

$$\begin{aligned}
\max \quad & \sum_{i,j} u_{ij}x_{ij} - \frac{(u_{ij}x_{ij})^2}{2C_j} \\
\text{s.t. for all } i, & \sum_j u_{ij}x_{ij} \le B_i \\
\text{and for all } j, & \sum_i x_{ij} \le 1. \\
& x_{ij} \ge 0.
\end{aligned}$$

The KKT conditions[3], that guarantee that a given pair of feasible solutions to the above programs are optimal, are

$$\begin{aligned}
& x_{ij} > 0 \Rightarrow p_j = u_{ij}(1 - \alpha_i - \theta_{ij}). \\
& p_j > 0 \Rightarrow \sum_i x_{ij} = 1. \\
& \alpha_i > 0 \Rightarrow \sum_j u_{ij}x_{ij} = B_i. \\
& \theta_{ij} = u_{ij}x_{ij}/C_j.
\end{aligned}$$

The restriction of the dual convex program to the sample S is defined as before. The algorithm now uses the α that minimizes this convex program.

$$\begin{aligned}
\min \quad & \sum_i \epsilon\alpha_i B_i + \sum_{j\in S} p_j + \sum_{i,j\in S} \frac{C_j}{2}\theta_{ij}^2 \qquad (2) \\
\text{s.t. } \forall\, i,\ \forall\, j \in S, \quad & p_j \ge u_{ij}(1 - \alpha_i - \theta_{ij}), \\
& \alpha_i, p_j \ge 0.
\end{aligned}$$

As before, the dual convex program is really a function of just the α_i's. This is because given the α_i's, one can separately solve for each j, the p_j's and the θ_{ij}'s by the following smaller convex program.

$$\begin{aligned}
\min \quad & p_j + \sum_i \frac{C_j}{2}\theta_{ij}^2 \qquad (3) \\
\text{s.t. } \forall\, i, \quad & p_j \ge u_{ij}(1 - \alpha_i - \theta_{ij}), \\
& p_j \ge 0.
\end{aligned}$$

This can be thought of as a smoothed version of $\max_i\{u_{ij}(1-\alpha_i)\}$. Once again, we can take the dual of (3), to get

$$\begin{aligned}
\max \quad & \sum_i u_{ij}(1 - \alpha_i)x_{ij} - \frac{(u_{ij}x_{ij})^2}{2C_j} \qquad (4) \\
\text{s.t. for all } j, & \sum_i x_{ij} \le 1. \\
& x_{ij} \ge 0.
\end{aligned}$$

We define $\widetilde{x}_{ij}(\alpha)$ to be the optimum solution to (4), which is the smoothed version of the earlier definition of $x_{ij}(\alpha)$. These define the allocations for a given α. Note that $\widetilde{x}_{ij}$ is uniquely defined by α. As earlier, we define the functions $\tilde{R}_i(\alpha) = \sum_j u_{ij}\widetilde{x}_{ij}(\alpha)$. Then $\tilde{P}_i(\alpha) = \min\{B_i, \tilde{R}_i(\alpha)\}$ is the profit extracted from bidder i.

[3] These are a generalization of the LP complementary slackness conditions to convex programs.

Algorithm 4.2: LEARN-WEIGHTS-SMOOTHED(ϵ)

for $j \leftarrow 1$ **to** ϵm
  Observe the bids u_{ij}.
  Allocate item j arbitrarily (e.g. all $x_{ij} = 0$).
Let α^* be the minimizer of Convex Program (2)
for $j \leftarrow \epsilon m + 1$ **to** m
  Observe the bids u_{ij}.
  Give $\widetilde{x}_{ij}(\alpha^*)$ fraction of item j to bidder i.

Define $\tilde{D}_i(\alpha) = \alpha_i B_i + (1-\alpha_i)\tilde{R}_i(\alpha)$. Notice that $\tilde{D}(\alpha) = \sum_i \tilde{D}_i(\alpha)$ is not the dual objective function. They differ by $\sum_i \frac{(u_{ij}x_{ij}(\alpha))^2}{2C_j}$. We define $\tilde{D}$ this way so that the relation between $\tilde{D}_i$, B_i and $\tilde{R}_i$ is the same as that for the non-smoothed versions. Also note that the KKT conditions are essentially the same as the complementary conditions with the smoothed versions.

However, the relation that $D(\alpha) \ge \text{OPT}$ is no longer true with the smoothed version. We now show that if we pick C_j appropriately, then an approximate version of the same relation is true. Let $C_j = \frac{\max_i\{u_{ij}\}}{2\epsilon}$. Then $\sum_i \frac{(u_{ij}x_{ij})^2}{2C_j} \le \epsilon\sum_i u_{ij}x_{ij}$. This implies that $\tilde{D}(\alpha) \ge (1-\epsilon)\text{OPT}$ for all α. With this, the competitive analysis of the algorithm goes through almost unchanged, with the smoothed versions of the functions replacing their original versions, with the introduction of an extra ϵOPT.

We now argue that if Λ is a grid of size $\delta = \frac{\epsilon^2}{\lambda^2}$, then moving α to the nearest point in Λ changes $\widetilde{x}_{ij}(\alpha)$ by at most $O(\epsilon)$. Note that for any given query j, moving α to the nearest point in Λ changes the optimum solution to (3) itself by $O(\delta \max_i\{u_{ij}\})$. This implies that the change in θ_{ij} is $O(\delta \max_i\{u_{ij}\}/u_{ij})$ for any i. Since $\widetilde{x}_{ij} = \frac{C_j\theta_{ij}}{u_{ij}}$, we get that the change in $\widetilde{x}_{ij}$ is at most

$$O\left(\frac{C_j\delta\max_i\{u_{ij}\}}{u_{ij}^2}\right) = O\left(\frac{\delta(\max_i\{u_{ij}\})^2}{\epsilon u_{ij}^2}\right),$$

which is less than ϵ if $\delta = \frac{\epsilon^2}{\lambda^2}$. Thus Λ is an $(\tilde{x}, \epsilon)$-net and $\log(|\Lambda|) = O(n\log(\lambda/\epsilon))$. This along with Lemma 5 implies Theorem 1.

5. CONCLUSION AND OPEN PROBLEMS

We solve an open problem of [19] by giving a $1-o(1)$ competitive algorithm for the adwords problem under a random permutation. Our result further opens other problems. In particular,

- Can the guarantee required in Theorem 1 be improved to require a smaller dependence on n and ϵ? The current algorithm itself might need a weaker guarantee.
- Can similar results be obtained with a richer model, incorporating other aspects of sponsored search, such as, multiple slots, pay-per-click schemes, second price payments, and so on?
- Can similar results be obtained for alternate objective functions? In particular, are there natural measures of *fairness* that can be approximated well?

- The random permutation model is applicable to other online problems as well. What other impossibilities in the worst case can be circumvented by going to the random permutation model?

6. REFERENCES

[1] Gagan Aggarwal, Ashish Goel, and Rajeev Motwani. Truthful auctions for pricing search keywords. In *EC '06: Proceedings of the 7th ACM conference on Electronic commerce*, pages 1–7, 2006.

[2] Moshe Babaioff, Nicole Immorlica, and Robert Kleinberg. Matroids, secretary problems, and online mechanisms. In *SODA '07: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 434–443, 2007.

[3] Maria-Florina Balcan, Avrim Blum, Jason D. Hartline, and Yishay Mansour. Mechanism design via machine learning. In *FOCS*, pages 605–614, 2005.

[4] Christian Borgs, Jennifer Chayes, Nicole Immorlica, Mohammad Mahdian, and Amin Saberi. Multi-unit auctions with budget-constrained bidders. In *EC '05: Proceedings of the 6th ACM conference on Electronic commerce*, pages 44–51, 2005.

[5] Niv Buchbinder, Kamal Jain, and Joseph Naor. Online primal-dual algorithms for maximizing ad-auctions revenue. In *ESA*, pages 253–264, 2007.

[6] E. B. Dynkin. The optimum choice of the instant for stopping a markov process. *Sov. Math. Dokl.*, 4, 1963.

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

[9] Jon Feldman, S Muthukrishnan, Martin Pal, and Cliff Stein. Budget optimization in search-based advertising auctions. In *EC '07: Proceedings of the 8th ACM conference on Electronic commerce*, pages 40–49, 2007.

[10] Gagan Goel and Aranyak Mehta. Online budgeted matching in random input models with applications to adwords. In *SODA '08: Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 982–991, 2008.

[11] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

[12] Bala Kalyanasundaram and Kirk R. Pruhs. An optimal deterministic algorithm for online b-matching. *Theor. Comput. Sci.*, 233(1-2):319–325, 2000.

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

[14] Robert Kleinberg. A multiple-choice secretary algorithm with applications to online auctions. In *SODA '05: Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 630–631, Philadelphia, PA, USA, 2005. Society for Industrial and Applied Mathematics.

[15] Erik Krohn and Kasturi Varadarajan, 2007. Private Communication.

[16] S. Lahaie, D. Pennock, A. Saberi, and R. Vohra. *Algorithmic Game Theory*, chapter Sponsored Search. Cambridge University Press, 2007.

[17] Sébastien Lahaie. An analysis of alternative slot auction designs for sponsored search. In *EC '06: Proceedings of the 7th ACM conference on Electronic commerce*, pages 218–227, 2006.

[18] Mohammad Mahdian, Hamid Nazerzadeh, and Amin Saberi. Allocating online advertisement space with unreliable estimates. In *EC '07: Proceedings of the 8th ACM conference on Electronic commerce*, pages 288–294, 2007.

[19] Aranyak Mehta, Amin Saberi, Umesh Vazirani, and Vijay Vazirani. Adwords and generalized online matching. *J. ACM*, 54(5):22, 2007.

[20] Hal R. Varian. Position auctions. *International Journal of Industrial Organization*, 25(6):1163–1178, December 2007.