AdWords and Generalized On-line Matching

Aranyak Mehta Amin Saberi Umesh Vazirani Vijay Vazirani

Abstract

How does a search engine company decide what ads to display with each query so as to maximize its revenue? This turns out to be a generalization of the online bipartite matching problem. We introduce the notion of a tradeoff revealing LP and use it to derive two algorithms achieving competitive ratios of $1 - 1/e$ for this problem.

1 Introduction

Internet search engine companies, such as Google, Yahoo and MSN, have revolutionized not only the use of the Internet by individuals but also the way businesses advertise to consumers. Instead of flooding consumers with unwanted ads, search engines open up the possibility of a dialogue between consumers and businesses, with consumers typing in keywords, called Adwords by Google, that reveal what they are looking for and search engines displaying highly targeted ads relevant to the specific query.

The AdWords market[1] is essentially a large auction where businesses place bids for individual keywords, together with limits specifying their maximum daily budget. The search engine company earns revenue from businesses when it displays their ads in response to a relevant search query (if the user actually clicks on the ad). Indeed, most of the revenues of search engine companies are derived in this manner[2].

In this context, the following computational problem, which we call the Adwords problem, was recently posed by Henzinger [5]: assign user queries to advertisers to maximize the total revenue. Observe that the task is necessarily online – when returning results of a specific query, the search engine company needs to immediately determine what ads to display on the side.

It is easy to see that the competitive ratio of the greedy algorithm is 1/2; moreover, this is tight. In this paper, we present two algorithms, one deterministic and one randomized, achieving competitive ratios of $1 - 1/e$ for this problem. The first algorithm needs to keep track of the money spent by each advertiser, but the second one does not and is therefore useful if the search engine company is using a distributed set of servers which periodically coordinate the money spent by each advertiser. In Section 7.2 we show that no algorithm, deterministic or randomized, can achieve a better competitive ratio.

In Section 6 we show how our algorithm and analysis can be generalized to the following, more realistic, situations, while still maintaining the same competitive ratio:

[1] This market dwarfs the AdSense market where the ad is based on the actual contents of the website.

[2] According to a recent New York Times article (Feb 4, 2005), the revenue accrued by Google from this market in the last three months of 2004 alone was over a billion dollars.

- A bidder pays only if the user clicks on his ad.
- Advertisers have different daily budgets.
- Instead of charging a bidder his actual bid, the search engine company charges him the next highest bid.
- Multiple ads can appear with the results of a query.
- Advertisers enter at different times.

1.1 Previous work

The adwords problem is clearly a generalization of the online bipartite matching problem: the special case where each advertiser makes unit bids and has a unit daily budget is precisely the online matching problem.

Even in this special case, the greedy algorithm achieves a competitive ratio of 1/2. least The algorithm that allocates each query to a random interested advertiser does not do much better – it achieves a competitive ratio of $1/2 + O(\log n/n)$.

In [10], Karp, Vazirani and Vazirani gave a randomized algorithm for the online matching problem achieving a competitive ratio of $1 - 1/e$. In their algorithm they fix a random permutation in advance and break ties according to that permutation. They further showed that no randomized online algorithm can achieve a better competitive ratio.

In another direction, Kalyanasundaram and Pruhs [9] considered the online b-matching problem which can be described as a special case of the adwords problem as follows: each advertiser has a daily budget of b dollars, but makes only 0/1 dollar bids on each query. Their online algorithm awards the query to that interested advertiser who has the highest left-over money. They show that the competitive ratio of this algorithm tends to $1 - 1/e$ as b tends to infinity. They also prove a lower bound of $1 - 1/e$ for deterministic algorithms.

Note that the adwords problem is NP-hard in the offline case. The best known approximation algorithm for this problem [1] gives an approximation factor of $1 - 1/e$, by randomized rounding. For our application, we assume throughout that the ratio of the bids to the budgets is sufficiently small.

1.2 Our results

To generalize the algorithms of [9] (resp. [10]) to arbitrary bids, it is instructive to examine the special case with bids restricted to $\{0, 1, 2\}$. The natural algorithm to try assigns each query to a highest bidder, using the previous heuristics to break ties (largest remaining budget/ highest ranking in the random permutation). We give examples (in the Appendix) showing that both these algorithms achieve competitive ratios strictly smaller and bounded away from $1 - 1/e$.

This indicates the need to consider a much more delicate tradeoff between the bid versus the remaining budget in the first case, and the bid versus the position in the random permutation in the second. The correct tradeoff function is derived by a novel LP-based approach, which

we outline below. The resulting algorithm is very simple, and is based on the following tradeoff function:

$$\psi(f) = 1 - e^{-(1-f)}$$

We provide two algorithms which achieve a factor of $1 - 1/e$:

Algorithm 1:

Allocate the next query to the bidder i maximizing the product of his bid and $\psi(T(i))$, where $T(i)$ is the fraction of the bidder's budget which has been spent do far, i.e. $T(i) = \frac{m_i}{B_i}$, where B_i is the total budget of bidder i, m_i is the amount of money spent by bidder i.

Algorithm 2:

Start by permuting the advertisers at random. Allocate the next query to the bidder maximizing the product of his bid and $\psi(i/n)$, where i is the rank of this bidder in the random order and n is the number of bidders.

Both algorithms assume that the daily budget of advertisers is large compared to their bids.

We now outline how we derive the correct tradeoff function. For this we introduce the notion of a *tradeoff-revealing family of LP's*, which extends the notion of a *factor-revealing LP*. The latter notion was implicit in [13, 4, 12] and was formalized and made explicit in [7, 6].

We start by providing a simpler proof of the Kalyanasundaram and Pruhs [9] result using a factor-revealing LP. We give an LP, L, whose constraints are satisfied at the end of a run of BALANCE on any instance π of the 0/1 adwords problem. The objective function of L gives the performance of BALANCE on π. Hence the optimal objective function value (over all instances π) of L is a lower bound on the competitive ratio of BALANCE. How good is this lower bound? Clearly, this depends on the constraints we have captured in L. It turns out that the bound computed by our LP is $1 - 1/e$ which is tight. Indeed, for some fairly sophisticated algorithms, e.g., [6, 2] a factor-revealing LP is the only way known of deriving a tight analysis.

Next, we obtain the function ψ as follows for the case of arbitrary bids. For each instance π, we provide a family of LP's $L(\pi, \psi)$, one corresponding to each decreasing tradeoff function ψ. Let $D(\pi, \psi)$ be the dual of $L(\pi, \psi)$. The objective function of $L(\pi, \psi)$ gives the performance of Algorithm 1 when run on π with tradeoff function ψ. The problem now is to choose ψ that yields a performance of at least $1 - 1/e$.

For this, we use an idea inspired by sensitivity analysis. It turns out that $L(\pi, \psi)$ differs from L only in that a vector $\Delta(\pi, \psi)$ is added to the right hand side of the constraints. Sensitivity analysis tells us that the optimal solution to $L(\pi, \psi)$ differs from that of L by $\Delta \cdot y^*$, where y^* is the optimal solution to the dual D of L, provided Δ is small. Using additional properties of our LPs, we show that this holds for arbitrarily large perturbations Δ. Next, we show that if y^* itself is used to define ψ in a specific manner, then $\Delta \cdot y^* \leq 0$. Observe that this function ψ does not depend on π and hence it works for all instances, giving a competitive ratio of $1 - 1/e$

for Algorithm 1. We call this ensemble $L(\pi, \psi)$ a *tradeoff revealing family of LP's.*

2 Problem Definition

There are N bidders each with a specified budget. Let B_i be the budget of bidder i. There is a set of queries arriving in an online fashion. Bidder i bids an amount of $b_{ij} \geq 0$ for query j. When a query arrives, the algorithm can assign it to one of the bidders and charge him an amount equal to the minimum of his bid for this query and his remaining budget. The objective is to maximize the revenue or the total amount of money charged.

Note that the above problem is NP-hard even in the off-line case. The best known approximation algorithm for this problem gives an approximation factor of $1 - \frac{1}{e}$, by a randomized rounding of the solution of the corresponding linear program [1]. For our applications, it is reasonable to assume that the ratio of the bids to the budgets is sufficiently small. We will make this assumption throughout the paper.

We say an algorithm is α-competitive if the ratio of the revenue of this algorithm over the revenue of the best off-line algorithm over all sequences of input is at least α.

In the next section we present an algorithm which achieves an allocation with revenue at least $1 - \frac{1}{e}$ of the revenue of the best possible allocation, and hence has a competitive factor of $1 - \frac{1}{e}$ for the above problem.

3 A Deterministic Algorithm

To the best of our knowledge, the algorithm used by most of the search engines is greedy. The greedy algorithm allocates every query to the bidder from whom it can charge the maximum amount of money (i.e. a bidder who has the highest value of min{bid, remaining budget}). It is easy to see that this algorithm achieves a factor of $\frac{1}{2}$ (see, e.g., [11]). One can also construct a scenario in which the greedy algorithm performs as bad as half even for the case where there are only two bidders: Suppose both bidders have the same budget B. In the first round, a certain number of queries arrive, for each of which the first bidder bids a bid of b, and the second bidder bids a bid of $b + \epsilon$, for some small $\epsilon > 0$. There will be $\frac{B}{b+\epsilon}$ such queries in the first round. The greedy algorithm gives all these queries to the second bidder and its budget is exhausted. Now a large number of queries arrive for which only the second bidder bids, and hence these queries are all wasted by the greedy algorithm. Clearly, the greedy algorithm makes a revenue of B, while the revenue of the best allocation is close to $2B$.

The main drawback of the greedy algorithm is that it is exhausting the budget of the first bidder even though he is paying only slightly higher for the queries. The idea of our first algorithm is that to determine the importance of a bidder for a query, we have to take into consideration his bid as well as the amount of money he has remaining. In fact, we will look at a tradeoff function of the bid and the remaining money of the bidders for every assignment.

Let us assume that each bidder has a budget of 1. We also assume that in the optimal allocation of the queries, every bidder spends all his money. These assumptions are without

loss of generality, and our proofs can be easily generalized to work without these assumptions. We pick a large enough integer k, and discretize the budget of a bidder into k parts of size $1/k$ each.

Definition: At any time during the run of the algorithm, we say that the *current type* of a bidder is i if it is filled between $(i-1)/k$ and i/k at that time. We say that the *final type* of a bidder is i if it is filled between $(i-1)/k$ and i/k at the end of the algorithm.

Let $\psi : [1 \ldots k] \rightarrow \mathbf{R}^+$ be the following (monotonically decreasing) function:

$$\psi(i) = 1 - \left(1 - \frac{1}{k}\right)^{k-i+1}$$

We say that $\psi(i)$ is the weight given to type i. If the current type of a bidder is i, then its current weight is $\psi(i)$. Our algorithm works as follows:

Algorithm 1

1. When a new query arrives, let the bid of bidder i be $b(i)$.
2. Let the current type of bidder i be $T(i)$.
3. Allocate the query to the bidder i who has the highest product $b(i) \times \psi(T(i))$.

Note that in the special case when all the bids are equal, our algorithm works in the same way as the BALANCE algorithm of [9], for any monotonically decreasing tradeoff function. This algorithm assigns each query to the interested bidder who has the most remaining money.

In the next section we prove that the competitive ratio of Algorithm 1 is $1 - 1/e$.

4 Analysis of Algorithm 1

In this section we prove that the competitive ratio of Algorithm 1 is $1 - 1/e$. This proof does not provide any intuition about the choice of the tradeoff function ψ. In Section 5, we introduce the LP based approach that led to the discovery of the correct tradeoff function ψ. In the process, this LP based approach also provides a proof that the competitive ratio of Algorithm 1 is $1 - 1/e$.

Theorem 1 *The competitive ratio of Algorithm 1 is* $1 - 1/e$.

Proof : For $i = 1, .., k-1$, define x_i to be the number of bidders who have spent within $[\frac{i-1}{k}, \frac{i}{k})$ fraction of their budgets at the end of the algorithm. Since all bidders have a budget of 1, and they spend all their money in OPT, we may also interpret x_i as the total amount of money spent by such bidders in OPT.

For $i = 1, .., k$, define w_i to be the amount of money spent by all the bidders from the interval $[\frac{i-1}{k}, \frac{i}{k})$ of their budgets. If a bidder spends more than $\frac{i}{k}$ amount of money at the end of the algorithm, then his contribution to w_i is $1/k$; if he spends less than $\frac{i-1}{k}$ then his contribution is 0; otherwise his contribution is the money he spends minus $\frac{i-1}{k}$.

The following equations are immediate from the definitions:

$$\forall i: \quad w_i = \frac{N - \sum_{j=1}^{i} x_j}{k} \tag{1}$$

Consider a query q which enters at time t. Suppose OPT gives q to a bidder who has spent within $[\frac{j-1}{k}, \frac{j}{k})$ at time t (current type j), and has spent within $[\frac{j'-1}{k}, \frac{j'}{k})$ at the end of the algorithm (final type j'). Suppose the algorithm gives q to a bidder who has spent within $[\frac{i-1}{k}, \frac{i}{k})$ at time t (current type i). Let opt_q be the amount of money that OPT gets for q, and let alg_q be the amount of money that the algorithm gets for q. Then, by the rule of the algorithm, we have

$$\psi(j) opt_q \ \leq \ \psi(i) alg_q$$

Also $j' \geq j$, hence $\psi(j') \leq \psi(j)$. So we have:

$$\psi(j') opt_q \ \leq \ \psi(i) alg_q \tag{2}$$

Now we sum over all queries q. For each query, the left side of (2) contributes to the sum $\sum_i \psi(i) x_i$, and the right side contributes to $\sum_i \psi(i) w_i$. Thus we get:

$$\sum_i \psi(i) x_i \ \leq \ \sum_i \psi(i) w_i \tag{3}$$

Plug in for w_i from (1), and plug in the choice of ψ:

$$\psi(i) = 1 - (1 - \frac{1}{k})^{k-i+1}$$

We get:

$$\sum_{i=1}^{k} x_i \frac{k-i+1}{k} \leq \frac{N}{e} \tag{4}$$

But the left side of (4) is precisely the amount of money left unspent at the end of the algorithm. Hence the factor of the ψ-based algorithm is at least $1 - 1/e$. □

5 A Tradeoff Revealing LP

In this section we describe the LP based approach which was used to derive the correct tradeoff function ψ. In the process we also bound the competitive ratio of the resulting algorithm, thus showing that the competitive ratio of Algorithm 1 is $1 - 1/e$.

We start with the special case in which all non-zero bids are equal. For any monotonically decreasing function ψ, Algorithm 1 reduces to the algorithm BALANCE of [9], in which each

query is given to the interested bidder who has the maximum unspent budget. We can write an LP L whose inequalities describe constraints on the state at the end of the algorithm, and whose objective function is 1 minus the competitive ratio:

$$\begin{aligned} \text{maximize} \quad & \sum_{i=1}^{k-1} \tfrac{k-i}{k} x_i \\ \text{subject to} \quad & \forall 1 \le i \le k: \ \sum_{j=1}^{i} (1 + \frac{i-j}{k}) x_j \le \frac{i}{k} N \\ & \forall 1 \le i \le k: \ x_i \ge 0 \end{aligned}$$

Here k is a sufficiently large discretization parameter, and for $i = 1, 2, \ldots, k-1$, x_i is the number of bidders who spent between $(i-1)/k$ and i/k of their budget at the end of algorithm. Lemma 2 shows that this LP L has the claimed properties. Such linear programs are called factor-revealing LPs and they have been used for the analysis of approximation algorithms and also online algorithms previously [6, 2]. This factor-revealing LP will reveal a factor of $1 - 1/e$ for the special case.

In the general case of arbitrary bids, two questions have to be answered:

1. What is the best tradeoff function ψ?
2. What is the worst input for the algorithm based on this ψ?

To answer these, we will first show how a rule based on any fixed tradeoff function ψ modifies the LP L obtained in the special case. Let us represent L as:

$$\text{Maximize } c \cdot x$$

$$\text{s.t.: } \mathbf{A}x \le b$$

Then, for each instance π of the problem, we get an LP $L(\pi, \psi)$ for the ψ-based algorithm. It can be shown that $L(\pi, \psi)$ has the form:

$$\text{Maximize } c \cdot x$$

$$\text{s.t.: } \mathbf{A}x \le b + \Delta(\pi, \psi)$$

where $\Delta(\pi, \psi)$ is a vector which depends on the ψ-rule. Δ keeps a count of how the ψ-based rule changes the allocation that a simple Balance algorithm would find - with the ψ-based rule, a query may go to a bidder who has smaller remaining budget, if that bidder bids high enough.

If Δ was a small perturbation, then standard sensitivity analysis tells us that the optimum value changes with Δ at a rate which is determined by the dual optimum solution y^*. In particular, the change in the objective function is approximately $\Delta \cdot y^*$. We now introduce two new ideas: First, we show that for our LP, the dual optimal solution y^* remains unchanged for Δ determined by any ψ rule (even though the Δ may be a large change). This means that the

change in the optimal objective value is precisely $\Delta \cdot y^*$. Secondly, we choose a ψ rule which itself depends on y^*, say $\psi = f(y^*)$. Since $\Delta(\pi, \psi)$ is determined by the ψ rule, we are able to choose $\psi = f(y^*)$ in such a way as to make $\Delta(\pi, f(y^*)) \cdot y^* \leq 0$. This means that the optimal objective value does not increase from the special case.

This gives answers to both the questions: The best ψ rule is as determined above. The worst case factor is no worse than the factor for the simple case, namely $1 - 1/e$. Thus our ensemble of LPs $L(\pi, \psi)$ is now not just factor revealing but also reveals the best tradeoff curve for the algorithm.

5.1 The case of equal bids

We pick a parameter k, which is an integer that we will take to be sufficiently large. For $i = 1, 2, \ldots, k-1$, let x_i be the number of bidders who spent between $(i-1)/k$ and i/k money at the end of the algorithm (i.e. of final type i). We will denote by ALG the revenue generated in Algorithm 1, as well as the algorithm itself.

Lemma 2

$$\forall\ i,\ 1 \leq i \leq k-1: \qquad \sum_{j=1}^{i} (1 + \frac{i-j}{k}) x_j \leq \frac{i}{k} N$$

Proof : For every i, consider the $x_1 + x_2 + \cdots + x_i$ bidders which have spent at most $\frac{i}{k}$ of their budget at the end of ALG, and look at the queries that OPT allocated to these bidders. These queries can only be allocated to bidders who have spent at most i/k of their money at the time of allocation. Therefore the total money to which these queries can be allocated is at most $\frac{i}{k}N$ minus the amount of money that was not spent because some bidders spent strictly less that i/k of their budget. Therefore we have:

$$\sum_{j=1}^{i} x_j \ \leq\ \frac{i}{k} N - \sum_{j=1}^{i} (\frac{i-j}{k}) x_j$$

□

At the end of ALG there are x_i bidders who have spent in between $\frac{i-1}{k}$ and $\frac{i}{k}$, for $i = 1, ..., k-1$ of their budget. We will overcount and consider them to have spent exactly $\frac{i}{k}$. There are $N - \sum_{i=1}^{k-1} x_i$ bidders who have spent more than $\frac{k-1}{k}$. We will consider these to have spent all their money. The total overcounting is at most $N\frac{1}{k}$.

Hence the revenue of the algorithm is

$$\begin{aligned} ALG\ &\geq\ \sum_{i=1}^{k-1} \frac{i}{k} x_i + \left(N - \sum_{i=1}^{k-1} x_i \right) - \frac{N}{k} \\ &=\ N - \sum_{i=1}^{k-1} \frac{k-i}{k} x_i - \frac{N}{k} \end{aligned}$$

To find the worst case performance of the algorithm we want to find the minimum value that $N - \sum_{i=1}^{k-1} \frac{k-i}{k} x_i - \frac{N}{k}$ can take over the feasible $\{x_i\}s$. This gives the following LP, which we call L:

$$\begin{aligned} \text{maximize} \quad & \Phi = \sum_{i=1}^{k-1} \tfrac{k-i}{k} x_i \\ \text{subject to} \quad & \forall 1 \le i \le k: \ \sum_{j=1}^{i} (1 + \frac{i-j}{k}) x_j \le \frac{i}{k} N \\ & \forall 1 \le i \le k: \ x_i \ge 0 \end{aligned} \tag{5}$$

Let us also write down the dual LP, which we will use in the case of arbitrary bids.

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{k-1} \tfrac{i}{k} N y_i \\ \text{subject to} \quad & \forall 1 \le i \le k-1: \ \sum_{j=i}^{k-1} (1 + \frac{j-i}{k}) y_j \ge \frac{k-i}{k} \\ & \forall 1 \le i \le k-1: \ y_i \ge 0 \end{aligned}$$

Lemma 3 *As $k \to \infty$, the value Φ of the above linear program goes to $\frac{N}{e}$*

Proof : One can easily verify that the above LP will be optimized when

$$x_i = \tfrac{N}{k}(1 - \tfrac{1}{k})^{i-1}, \quad \text{for } i = 1, .., k-1$$

This gives an objective function value of

$$\begin{aligned} \Phi &= \sum_{i=1}^{k-1} (\tfrac{k-i}{k}) \tfrac{N}{k} (1 - \tfrac{1}{k})^{i-1} \\ &= N(1 - \tfrac{1}{k})^k \end{aligned}$$

As we make the discretization finer (i.e. as $k \to \infty$) Φ tends to $\frac{N}{e}$.

□

Recall that the size of the matching is at least $N - \Phi - \frac{N}{k}$, hence it tends to $N(1 - \frac{1}{e})$. Since OPT is N, the competitive ratio is at least $1 - \frac{1}{e}$. On the other hand one can use the LP to find an instance of the problem such at the end of the algorithm all the inequalities of the primal are tight, hence the competitive ratio of ALG is exactly $1 - \frac{1}{e}$.

5.2 Arbitrary bids

We start with an arbitrary monotonically decreasing tradeoff function ψ, and the given instance π of the adwords problem. Consider the time when query q enters. Let A_q be the bidder to which ALG allocates q and let alg_q be the bid of A_q. Let B_q be the bidder to which OPT allocates q, and let opt_q be the bid of B_q. Let the current type of A_q be $T(A_q)$ and the current type of B_q be $T(B_q)$. Let the final type of B_q be $F(B_q)$. By the rule of the algorithm, we have

$$\psi(T(A_q))alg_q \;\geq\; \psi(T(B_q))opt_q$$

Since $F(B_q) \geq T(B_q)$, and ψ is monotonically decreasing, we have

$$\psi(T(A_q))alg_q \;\geq\; \psi(F(B_q))opt_q \tag{6}$$

For the given instance π and the tradeoff function ψ, take all the queries that OPT assigns to a bidder of final type i, and sum up the money that OPT gets from these queries. Call this α_i. Take all the queries that ALG allocates to a bidder of current type i, and sum up all the money that ALG gets from these queries. Call this β_i. Define $\gamma_i := \alpha_i - \beta_i$.

Lemma 4 *For any instance of the adwords problem and for any decreasing function ψ,*

$$S := \sum_{i=1}^{k-1} \gamma(i)\psi(i) \leq 0.$$

Proof : Every positive contribution to γ_i (from α_i) corresponds to a query q that OPT gave to a bidder of final type i. Suppose ALG gave q to a bidder of current type j. Then q also provides a negative contribution to γ_j (from β_j). But by the rule of the algorithm (equation 6), we know that $\psi(j)alg_q \geq \psi(i)opt_q$. Thus the negative contribution of this query q to $\psi(j)\gamma_j$ is at least the positive contribution of q to $\psi(i)\gamma_i$, giving a net non-positive contribution to S. □

With these definitions one can see, using an argument similar to the one in Lemma 1, that the run of ALG on this instance satisfies the following equality:

Lemma 5

$$\forall\ 1 \leq i \leq k-1 : \quad \sum_{j=1}^{i}(1 + \frac{i-j}{k})x_j \;=\; \frac{i}{k}N + \sum_{j=1}^{i}\gamma_j$$

The revenue of ALG remains $N - \sum_{i=1}^{k-1} \frac{k-i}{k}x_i$. In order to find the best function ψ and put a lower bound on the revenue, we will relax this equality in Lemma 5 to an inequality and consider the following LP and its dual. We denote the primal by $L(\pi, \psi)$, to emphasize that it depends on the problem instance π and the tradeoff function ψ.

$$\begin{aligned} \text{maximize} \quad & \sum_{i=1}^{k-1} \tfrac{k-i}{k}x_i \\ \text{subject to} \quad & \forall 1 \leq i \leq k-1 : \ \sum_{j=1}^{i}(1 + \frac{i-j}{k})x_j \;\leq\; \frac{i}{k}N + \sum_{j=1}^{i}\gamma_j \\ & \forall 1 \leq i \leq k-1 : \ x_i \geq 0 \end{aligned} \tag{7}$$

The dual LP is:

$$\begin{aligned}
\text{minimize} \quad & \sum_{i=1}^{k-1}(\tfrac{i}{k}N + \sum_{j=1}^{i}\gamma_j)y_i \\
\text{subject to} \quad & \forall 1 \le i \le k-1: \ \sum_{j=i}^{k-1}(1+\frac{j-i}{k})y_j \ \ge \ \frac{k-i}{k} \\
& \forall 1 \le i \le k-1: \ y_i \ge 0
\end{aligned}$$

Observe that compared to LP (5), in our primal LP (7), only the right hand side changed. Therefore, the dual constraints remain unchanged, and only the dual objective function changes.

Let us call inequalities of the form $x_i \ge 0$ and $y_i \ge 0$ as trivial and the rest as non-trivial. The next lemma follows from the fact that all complementary slackness conditions are satisfied.

Lemma 6 *For any monotonically decreasing tradeoff function ψ, an optimal solution to the primal program and dual programs is obtained by setting all the non-trivial inequalities tight.*

As a result, the duals to LPs (5) and (7) have the same optimal solution, $y_i^* = \frac{1}{k}(1-\frac{1}{k})^{k-i-1}$. However, their objective function values differ because of the second term in the latter dual. Observe the similarity of this term and the sum S in Lemma 4, which is non-positive for any decreasing function ψ. Hence, we get,

Theorem 7 *For the ψ function defined as*

$$\psi(i) \ := \ \sum_{j=i}^{k-1} y_j^* \ = \ 1-(1-\frac{1}{k})^{k-i+1}$$

the competitive ratio of Algorithm 1 is $(1-\frac{1}{e})$.

Proof : By Lemma 6, the optimal dual objective function is:

$$\begin{aligned}
D &= \sum_{i=1}^{k-1}[\tfrac{i}{k}N + \sum_{j=1}^{i}\gamma_j]y_i^* \\
&= N(1-\tfrac{1}{k})^k + \sum_{j=1}^{k-1}\left(\sum_{j=1}^{i}\gamma_j\right)y_i^*
\end{aligned}$$

This consists of two parts: The first part is $N(1-\frac{1}{k})^k$ which goes to $\frac{N}{e}$ as the discretization gets finer. The second part is

$$\sum_{i=1}^{k-1}\left(\sum_{j=1}^{i}\gamma_j\right)y_i^* \ = \ \sum_{i=1}^{k-1}\gamma_i\left(\sum_{j=i}^{k-1}y_j^*\right) \ = \ \sum_{i=1}^{k-1}\gamma_i\psi(i) \ = \ S$$

Here the second equality is by our choice of ψ. By Lemma 4, S is non-positive. Hence the optimum is no more than $\frac{N}{e}$, and therefore the revenue of ALG is no less than $N(1-\frac{1}{e})$.

□

6 Towards more realistic models

In this section we show how our algorithm and analysis can be generalized to the following situations:

1. Advertisers have different daily budgets.
2. The optimal allocation does not exhaust all the money of advertisers
3. Advertisers enter at different times.
4. More than one ad can appear with the results of a query. The most general situation is that with each query we are provided a number specifying the maximum number of ads.
5. A bidder pays only if the user clicks on his ad.
6. A winning bidder pays only an amount equal to the next highest bid.

1, 2, 3: We say that the current type of a bidder at some time during the run of the algorithm is j if he has spent between $(j-1)/k$ and j/k fraction of his budget at that time. The algorithm allocates the next query to the bidder who maximizes the product of his bid and ψ(current type).

The proof of the competitive ratio changes minimally: Let the budget of bidder j be B_j. For $i = 1, .., k$, define w_i^j to be the amount of money spent by the bidder j from the interval $[\frac{i-1}{k}B_j, \frac{i}{k}B_j)$ of his budget. Let $w_i = \sum_j w_i^j$. Let u_i be the amount of money that the optimal allocation gets from the bins of final type i. Let $U = \sum_i u_i$, be the total amount of money obtained in the optimal allocation.

Now equation 1 of Section 4 becomes

$$\forall i: \quad w_i \geq \frac{U - \sum_{j=1}^{i} u_j}{k}$$

and equation 3 becomes

$$\sum_i \psi(i) u_i \;\leq\; \sum_i \psi(i) w_i$$

These two sets of equations suffice to prove that the competitive ratio is at least $1 - 1/e$. We also note that the algorithm and the proof of the competitive ratio remain unchanged even if we allow advertisers to enter the bidding process at any time during the query sequence.

4: If the arriving query q requires n_q number of advertisements to be placed, then allocate it to the bidders with the top n_q values of the product of bid and ψ(current type). The proof of the competitive ratio remains unchanged.

5: In order to model this situation, we simply set the effective bid of a bidder to be the product of his actual bid and his click-through rate (CTR), which is the probability that a user will click on his ad. We assume that the click-through rate is known to the algorithm in advance - indeed several search engines keep a measure of the click-through rates of the bidders.

6: So far we have assumed that a bidder is charged the value of his bid if he is awarded a query. Some search engine companies charge a lower amount: the next highest bid. There are two ways of defining "next highest bid": next highest bid for this query among all bids received at the start of the algorithm or only among alive bidders, i.e. bidders who still have money. It is easy to see that a small modification of our algorithm achieves a competitive ratio of $1 - 1/e$ for the first possibility: award the query to the bidder that maximizes next highest bid $\times$ ψ(fraction of money spent). Next, let us consider the second possibility. In this case, the offline algorithm will attempt to keep alive bidders simply to charge other bidders higher amounts. If the online algorithm is also allowed this capability, it can also keep all bidders alive all the way to the end and this possibility reduces to the first one.

7 A Randomized Algorithm

In this section we define a generalization of the RANKING algorithm of [10], which has a competitive ratio of $1 - 1/e$ for arbitrary bids, when the bid to budget ratio is small.

In this algorithm we pick a random permutation σ of the n bidders right at the beginning. For a bidder i, we call $\sigma(i)$ the position or rank bidder i in σ. Again, we choose the same tradeoff function to trade off the importance of the bid of a bidder and his rank in the permutation:

$$\psi(i) = 1 - \left(1 - \frac{1}{n}\right)^{n-i+1}$$

Algorithm 2:

1. Pick a random permutation σ of the bidders.
2. For each new query, let the bid of bidder i be $b(i)$.
3. Allocate this query to a bidder with the highest value of the product $b(i) \times \psi(\sigma(i))$.

7.1 Analysis of Algorithm 2

In this section we prove that the competitive ratio of Algorithm 2 is also $1 - 1/e$. We follow the direct proof of Section 4.

We first define the notion of a Refusal algorithm based on Algorithm 2, which will *disallocate* certain money from the bidders as follows. Refusal will run identically to Algorithm 2, with the following difference: Consider a query q which arrives in the online order. Let r_q be the bidder to whom OPT allocated q, and let opt_q be the amount of money that OPT gets for

q. Suppose that r_q has at least opt_q remaining budget when q arrives. Suppose further, that Refusal matches q to some bidder other than r_q (since this bidder has a higher product of bid and ψ-value). Then Refusal will disallocate opt_q money from r_q, i.e. it will artificially reduce the remaining budget of r_q by an amount opt_q.

Lemma 8 *All the money which is not disallocated is spent on queries.*

Lemma 9 *The competitive ratio of Refusal is at most the competitive ratio of Algorithm 2.*

We will now prove that the competitive ratio of Refusal is at least $1 - 1/e$.

Fix a query q and a permutation σ of the rows. Let r_q be the bidder to which OPT allocates q and let opt_q be the amount of money that OPT gets for q.

If Refusal matches q to r_q, then define $\alpha(q, \sigma) = n + 1$. Otherwise, we define $\alpha(q, \sigma)$ as follows: Let $A(q, \sigma)$ be the position in σ of the bidder to which Refusal matches q. Modify σ by shifting r_q upwards in the order, keeping the order of the rest of the bidders unchanged. Define $\alpha(q, \sigma)$ as the highest such position of r_q so that r_q has at least opt_q remaining budget when q arrives, and Refusal still matches q to the bidder in position $A(q, \sigma)$.

Define $x_i^q = opt_q \, Pr[\alpha(q, \sigma) = i]$, where the probability is taken over random σ. Let $x_i = \sum_q x_i^q$.

Define w_i^q to be the expected amount of money spent by the row in position i on query q. Let $w_i = \sum_q w_i^q$, the expected amount of money spent by the row in position i at the end of Refusal.

Lemma 10

$$\sum_i \psi(i) x_i \;\leq\; \sum_i \psi(i) w_i \tag{8}$$

Proof : Fix a query q and a permutation σ. Let r_q be the bidder to which OPT allocates q and let opt_q be the amount of money OPT gets for q. Let $A(q, \sigma)$ be the position in σ of the bidder to which Refusal matches q and let alg_q be the amount of money Refusal gets for q.

In the case that $\alpha(q, \sigma) \neq n + 1$, the following holds by the rule used by the algorithm:

$$\psi(\alpha(q, \sigma)) opt_q \;\leq\; \psi(A(q, \sigma)) alg_q$$

In the case that $\alpha(q, \sigma) = n + 1$, we simply write:

$$0 \;\leq\; \psi(A(q, \sigma)) alg_q$$

Taking expectation over random σ we get

$$\sum_i \psi(i) x_i^q \;\leq\; \sum_i \psi(i) w_i^q$$

Taking a summation over all queries q, we get

$$\sum_i \psi(i) x_i \;\leq\; \sum_i \psi(i) w_i$$

□

Lemma 11

$$\forall i: \quad w_i \ \geq \ 1 - \frac{\sum_{j=1}^{i} x_j}{n} \tag{9}$$

Proof : By Lemma 8, it is equivalent to prove that the expected amount of money disallocated in position i by Refusal is at most $\frac{\sum_{j=1}^{i} x_j}{n}$.

For a fixed query q and permutation σ, let r_q be the row to which OPT allocates q, and let $B(q,\sigma)$ be the position of r_q in σ. Then an opt_q amount of money is disallocated from r_q if and only if $\alpha(q,\sigma) \leq B(q,\sigma)$. In such a case, consider the following process. Start with a permutation derived from σ by shifting r_q to position $\alpha(q,\sigma)$. Replace r_q uniformly at random in each of the n positions. Then with probability $1/n$ we get back σ and opt_q amount of money is disallocated from r_q in position $B(q,\sigma)$. In this manner, we may only be overcounting the amount of disallocated money, since some of the positions for r_q below $\alpha(q,\sigma)$ may correspond to permutations σ' with a different (larger) value of $\alpha(q,\sigma')$.

Taking expectation over random σ and summing over all queries q, we get the statement of the lemma. □

Comparing to Section 4, we see that the constraints (8) and (9) are similar to the constraints obtained in that proof. The amount of money left unspent also has the same form, namely

$$\sum_{i=1}^{n}(1-w_i) \ \leq \ \sum_{i=1}^{n} x_i \frac{n-i+1}{n}$$

The only difference is that the equality in (1) has become an inequality in (9), but this does not change the proof. Hence we get

Proposition 12 *The competitive ratio of Refusal is at least* $1-1/e$.

From Lemma 9, we get:

Theorem 13 *The competitive ratio of the Algorithm 2 is at least* $1-1/e$.

7.2 A Lower Bound for Randomized Algorithms

In [10], the authors proved a lower bound of $1-1/e$ on the competitive ratio of any randomized online algorithm for the online bipartite matching problem. Also, [9] proved a lower bound of $1-1/e$ on the competitive ratio of any online deterministic algorithm for the online b-matching problem, even for large b. We show a lower bound of $1-1/e$ for for online randomized algorithms for the b-matching problem, even for large b. This also resolves an open question from [8].

Theorem 14 *No randomized online algorithm can have a competitive ratio better than* $1-1/e$ *for the b-matching problem, for large b.*

Proof : We use Yao's Lemma [14], which says that the worst case expected factor (over all inputs) of the best randomized algorithm is equal to the expected factor of the best deterministic algorithm for the worst distribution over inputs. Thus it suffices for our purpose to present a distribution over inputs such that any deterministic algorithm obtains at most $1 - 1/e$ of the optimal allocation on the average. By Yao's Lemma, this would put a bound of $1 - 1/e$ on the worst case performance of any randomized algorithm.

Consider first the worst case input for the algorithm BALANCE with N bidders, each with a budget of 1. In this instance, the queries enter in N rounds. There are $1/\epsilon$ number of queries in each round. We denote by Q_i the queries of round i, which are identical to each other. For every $i = 1, .., N$, bidders i through N bid ϵ for each of the queries of round i, while bidders 1 through $i - 1$ bid 0 for these queries. The optimal assignment is clearly the one in which all the queries of round i are allocated to bidder i, achieving a revenue of N. One can also show that BALANCE will achieve only $N(1 - 1/e)$ revenue on this input.

Now consider all the inputs which can be derived from the above input by permutation of the numbers of the bidders and take the uniform distribution $\mathcal{D}$ over all these inputs. Formally, $\mathcal{D}$ can be described as follows: Pick a random permutation π of the bidders. The queries enter in rounds in the order $Q_1, Q_2, ..., Q_N$. Bidders $\pi(i), \pi(i + 1), ..., \pi(N)$ bid ϵ for the queries Q_i and the other bidders bid 0 for these queries. The optimal allocation for any permutation π remains N, by allocating the queries Q_i to bidder $\pi(i)$. We wish to bound the expected revenue of any deterministic algorithm over inputs from the distribution $\mathcal{D}$.

Fix any deterministic algorithm. Let q_{ij} be the fraction of queries from Q_i that bidder j is allocated. We have:

$$E_\pi[q_{ij}] \leq \begin{cases} \frac{1}{N-i+1} & \text{if } j \geq i, \\ 0 & \text{if } j < i. \end{cases}$$

To see this, note that there are $N - i + 1$ bidders who are bidding for queries Q_i. The deterministic algorithm allocates some fraction of these queries to some bidders who bid for them, and leaves the rest of the queries unallocated. If $j \geq i$ then bidder j is a random bidder among the bidders bidding for these queries and hence is allocated an average amount of $\frac{1}{N-i+1}$ of the queries which were allocated from Q_i (where the average is taken over random permutations of the bidders). On the other hand, if $j < i$, then bidder j bids 0 for queries in Q_i and is not allocated any of these queries in any permutation.

Thus we get that the expected amount of money spent by a bidder j at the end of the algorithm is at most $\min\{1, \sum_{i=1}^{j} \frac{1}{N-i+1}\}$. By summing this over $j = 1, .., N$, we get that the expected revenue of the deterministic algorithm over the distributional input $\mathcal{D}$ is at most $N(1 - 1/e)$. This finishes the proof of the theorem. □

8 Discussion

Search engine companies accumulate vast amounts of statistical information which they do use in solving the Adwords problem. The main new idea coming from our study of this problem from the viewpoint of worst case analysis is the use of a tradeoff function. Blending this idea into the the currently used algorithms seems a promising possibility.

One may raise the objection that our algorithm is more favourable to people with larger budgets, since the fraction of their spent money is typically going to be smaller. For example, consider the following situation: the daily budget of bidder A is ten times that of B and they both bid equally for a certain keyword. Initially, our algorithm will favour A to B. However, observe that over an extended period, our algorithm will award one query to B each time it awards ten queries to A, and this certainly seems fair.

Several new issues arise: Our notion of tradeoff revealing family of LP's deserves to be studied further in the setting of approximation and online algorithms. Is it possible to achieve competitive ratio of $1-1/e$ when the budgets of advertisers are not necessarily large? As stated in the Introduction, both our algorithms assume that daily budgets are large compared to bids. It is worth noting that an online algorithm for this problem with a competitive ratio of $1-1/e$ will not only match the lower bound given in [10] for online algorithms but also the best known off-line approximation algorithm [1].

Finally, this new auctions setting seems ripe with new game theoretic issues. For example, some of the search engines (e.g., Google) use a mechanism similar to a second-price auction for charging the advertisers in order to achieve some degree of incentive compatibility. However, it seems that there are still various ways for the advertisers to game these mechanisms. Designing a truthful mechanism in this setting is an important open problem. Recently, [3] provided a partial answer for this problem by showing that under some assumptions, it is impossible to design a truthful mechanism that allocates all the keywords to budget constrained advertisers.

Acknowledgement: We would like to thank Monika Henzinger stating the Adwords problem and Milena Mihail, Serge Plotkin and Meredith Goldsmith for valuable discussions. We would also like to thank Subrahmanyam Kalyanasundaram for his help in finding the counterexample in Appendix B, and in implementing our algorithm.

References

[1] N. Andelman and Y. Mansour. Auctions with budget constraints. In *9th Scandinavian Workshop on Algorithm Theory (SWAT)*, pages 26–38, 2004.

[2] N. Bansal, L. Fleischer, T. Kimbrel, M. Mahdian, B. Schieber, and M. Sviridenko. Further improvements in competitive guarantees for QoS buffering. In *ICALP*, volume 3142 of *LNCS*, pages 196–207. Springer, 2004.

[3] Christian Borgs, Jennifer Chayes, Nicole Immorlica, Mohammad Mahdian, and Amin Saberi. Multi-unit auctions with budget-constrained bidders, 2004. Manuscript.

[4] M. Goemans and J. Kleinberg. An improved approximation algorithm for the minimum latency problem. *Mathematical Programming*, 82:111–124, 1998.

[5] Monika Henzinger. Personal communication.

[6] K. Jain, M. Mahdian, E. Markakis, A. Saberi, and V. Vazirani. Greedy facility location algorithms analyzed using dual fitting with factor-revealing lp. *J. ACM*, 2003.

[7] Kamal Jain, Mohammad Mahdian, and Amin Saberi. A new greedy approach for facility location problems. In *STOC*, pages 731–740, 2002.

[8] Bala Kalyanasundaram and Kirk Pruhs. On-line network optimization problems. In *Developments from a June 1996 seminar on Online algorithms*, pages 268–280. Springer-Verlag, 1998.

[9] Bala Kalyanasundaram and Kirk R. Pruhs. An optimal deterministic algorithm for online b -matching. *Theoretical Computer Science*, 233(1–2):319–325, 2000.

[10] R.M. Karp, U.V. Vazirani, and V.V. Vazirani. An optimal algorithm for online bipartite matching. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, 1990.

[11] B. Lehman, D. Lehman, and N. Nisan. Combinatorial auctions with decreasing marginal utilities. In *Proceedings of the 3rd ACM conference on Electronic Commerce*, pages 18 –28, 2001.

[12] M. Mahdian, E. Markakis, A. Saberi, and V. Vazirani. A greedy facility location algorithm analyzed using dual fitting. *RANDOM-APPROX*, pages 127–137, 2001.

[13] R.J. McEliese, E.R. Rodemich, H. Rumsey Jr., and L.R. Welch. New upper bounds on the rate of a code via the delsarte-macwilliams inequalities. *IEEE Trans. Inform. Theory*, pages 157–166, 1977.

[14] A. C. Yao. Probabilistic computations: towards a unified measure of complexity. *FOCS*, pages 222–227, 1977.

Appendix

A Counterexample 1

We present an example to show a factor strictly less than $1-1/e$ for the algorithm which gives a query to a highest bidder, breaking ties by giving it to the bidder with most left-over money. This example has only three values for the bids - 0, a or $2a$, for some small $a > 0$. Thus, in the case of arbitrary bids, the strategy of bucketing close enough bids (say within a factor of 2) together, and running such an algorithm does not work.

There are N bidders numbered $1, \ldots, N$, each with budget 1. We get the following query sequence and bidding pattern. Each bid is either 0, a or $2a$. Let $m = 1/a$. We will take $a \to 0$.

The queries arrive in N rounds. In each round m queries are made. The N rounds are divided into 3 phases.

Phase 1 ($1 \leq i \leq 0.4N$): In the first round m queries are made, for which the bidders $0.1N+1$ to N bid with a bid of a, and bidders 1 to $0.1N$ do not bid. Similarly, for $1 \leq i \leq 0.4N$, in

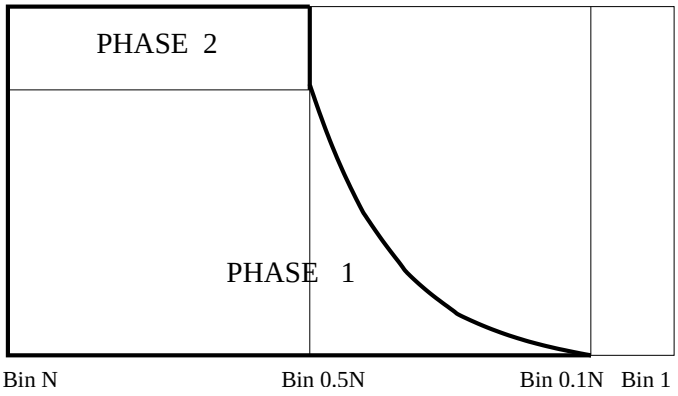

Figure 1: The bidders are ordered from right to left. The area inside the dark outline is the amount of money generated by the algorithm. The optimum allocation gets an amount equal to the whole rectangle.

the ith round m queries are made, for which bidders $0.1N + i$ to N bid with a bid of a, and for which bidders 1 to $0.1N + i - 1$ do not bid.

For $1 \leq i \leq 0.4N$, the algorithm will distribute the queries of the ith round equally between bidders $0.1N + i$ to N. This will give the partial allocation as shown in Figure A.

Phase 2 ($0.4N + 1 \leq i \leq 0.5N$): In the $(0.4N + 1)$th round m queries are made, for which bidder 1 bids a, and bidders $0.5N$ to N bid $2a$ (the rest of the bidders bid 0). Similarly, for $0.4N + 1 \leq i \leq 0.5N$, in the ith round m queries are made, for which bidder $i - 0.4N$ bids a, and bidders $0.5N$ to N bid $2a$.

For $0.4N + 1 \leq i \leq 0.5N$, the algorithm will distribute the queries of round i equally between bidders $0.5N$ to N.

At this point during the algorithm, bidders $0.5N + 1$ to N have spent all their money.

Phase 3 ($0.5N + 1 \leq i \leq N$): m queries enter in round i, for which only bidder i bids at a, and the other bidders do not bid.

The algorithm has to throw away these queries, since bidders $0.5N + 1$ to N have already spent their money.

The optimum allocation, on the other hand, is to allocate the queries in round i as follows:

- For $1 \leq i \leq 0.4N$, allocate all queries in round i to bidder $0.1N + i$.
- For $0.4N + 1 \leq i \leq 0.5N$, allocate all queries in round i to bidder $i - 0.4N$.
- For $0.5N + 1 \leq i \leq N$, allocate all queries in round i to bidder i.

Clearly, OPT makes N amount of money. A calculation shows that the algorithm makes $0.62N$ amount of money. Thus the factor is strictly less than $1 - 1/e$.

We can modify the above example to allow bids of 0, a and κa, for any $\kappa > 1$, such that the algorithm performs strictly worse that $1 - 1/e$.

As $\kappa \to \infty$, the factor tends to $1 - 1/e$, and as $\kappa \to 1$, the factor tends to $1/2$. Of course, if $\kappa = 1$, then this reduces to the original model of [9], and the factor is $1 - 1/e$.

B Counterexample 2

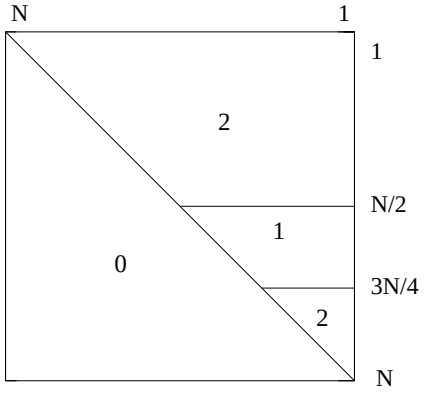

Figure 2: The rows N/2 to 3N/4 have a budget of 1. All other rows have a budget of 2.

The example consists of an $N \times N$ upper-triangular matrix, as shown in Figure B. The columns represent the queries and are ordered from right to left. The rows represent the bidders. The entry in the ith row and jth column is the bid of bidder i for query j. The entries in the upper triangle in rows 1 to $N/2$, and rows $3N/4$ to N are all 2. These bidders have a budget of 2. The entries in the upper triangle in rows $N/2 + 1$ to $3N/4$ are all 1. These bidders have a budget of 1.

The optimum allocation is along the diagonal, with column i allocated to row i. This generates $2N - N/4$ amount of money.

It can be proved that the algorithm gets $1.1N$ amount of money, which is strictly less than $1 - 1/e$ of the optimum.