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 an optimal algorithm achieving a competitive ratio 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. Typical search engine queries are short and reveal a great deal of information about user preferences. This gives search engine companies a unique opportunity to display highly targeted ads to the user.

The online advertising mechanisms used by search engines, including Google's AdWords, are essentially large auctions where businesses place bids for individual keywords, together with limits specifying their maximum daily budget. The search engine company earns revenue 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 [Bat05]. One factor in their dramatic success is that, unlike conventional advertising, search engine companies are able to cater to low budget advertisers (who occupy the fat tail of the power law distribution governing advertising budgets of companies and organizations).

The following computational problem, which we call the adwords problem, is a formalization of a question posed to us by Henzinger [Hen04]: There are N bidders, each with a specified daily budget b_i. Q is a set of query words. Each bidder i specifies a bid c_{iq} for query word $q \in Q$. A sequence $q_1 q_2 \ldots q_M$ of query words $q_j \in Q$ arrive online during the day, and each query q_j must be assigned to some bidder i (for a revenue of c_{iq_j}). The objective is to maximize the total revenue at the end of the day while respecting the daily budgets of the bidders.

*Google, Inc., 1600 Amphitheatre Parkway, Mountain View, CA.

†Department of Management Science and Engineering, Institute for Computational and Mathematical Engineering, Stanford University, saberi@stanford.edu. Supported by NSF Career Award and a gift from Google.

‡Computer Science Dept, U.C. Berkeley. vazirani@cs.berkeley.edu. Supported by NSF Grant 0635401, and NSF ITR Grant CCR-0121555.

§College of Computing, Georgia Institute of Technology. vazirani@cc.gatech.edu. Supported by NSF Grant 0728640.

In this paper, we present a deterministic algorithm achieving a competitive ratio of $1 - 1/e$ for this problem, under the assumption that bids are small compared to budgets. The algorithm is simple and time efficient. In Section 7 we show that no randomized algorithm can achieve a better competitive ratio, even under this assumption of small bids.

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:

- 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.

In practice there is additional statistical information available about search queries, and in Section 8 we discuss how to incorporate this additional information into our algorithm.

1.1 Online Bipartite Matching Algorithms

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. 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 [KVV90], Karp, Vazirani and Vazirani gave a randomized algorithm for the online matching problem achieving a competitive ratio of $1 - 1/e$. Their algorithm, called RANKING, fixes a random permutation of the bidders in advance and breaks ties according to their ranking in this permutation. They further showed that no randomized online algorithm can achieve a better competitive ratio.

In another direction, Kalyanasundaram and Pruhs [KP00] 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, called BALANCE, awards the query to that interested advertiser who has the highest unspent budget. 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.

1.2 Our Algorithm and Analysis Technique

It is easy to see that an algorithm that greedily assigns each query to the highest bidder achieves a competitive ratio of at most 1/2. Key to designing an optimal online algorithm is

finding the correct tradeoff between the bid and (fraction of) unspent budget. The tradeoff function used in our algorithm, which we derive by a novel LP-based approach, is the following:

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

The resulting algorithm is very simple:

Algorithm: 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 so 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 when the query arrives.

The algorithm assumes 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*. This concept builds on the notion of a *factor-revealing LP* [JMM$^+$03]. We start by writing a factor-revealing LP to analyze the performance in the special case when all bids are equal. This provides a simpler proof of the Kalyanasundaram and Pruhs [KP00] result.

We give an LP, L, whose constraints (upper bounding the number of bidders spending small fractions of their budgets) are satisfied at the end of a run of BALANCE on any instance π (sequence of queries) of the equal bids case. The objective function of L gives the performance of BALANCE on π. Hence the optimal objective function value 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., [JMM$^+$03, BFK$^+$04], a factor-revealing LP is the only way known of deriving a tight analysis.

Dealing with arbitrary bids is considerably more challenging, since we don't know how to write meaningful constraints reflecting the allocation of queries to bidders on an arbitrary instance π. The approach we use is rather counterintuitive. We proceed by fixing a monotonically decreasing tradeoff function ψ, as well as the sequence of queries π, and write a new LP $L(\pi, \psi)$ for the algorithm using tradeoff function ψ run on instance π. Of course, once we specify the algorithm as well as the sequence of queries, the actual allocation of queries to bidders is completely determined. $L(\pi, \psi)$ is identical to the factor revealing LP L except that the right hand side of each inequality is replaced by the actual value attained for this constraint in this run of the algorithm. How could these LP's $L(\pi, \psi)$ — whose inequalities are just relaxed tautologies with unknown right hand sides — possibly provide any non-trivial insight? It turns out that the family of LP's does capture some of the structure of the problem which is revealed by considering the family of dual linear programs $D(\pi, \psi)$.

Notice 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. Therefore, the dual programs $D(\pi, \psi)$ differ from the dual D of L only in the objective function, which is changed by $\Delta(\pi, \psi) \cdot \boldsymbol{y}$, where $\boldsymbol{y}$ is the vector of dual variables. Hence the dual polytope for all LP's in the family is the same as that for D. Moreover, we show that D and each LP in the family $D(\pi, \psi)$ attains its optimal value at the same vertex, $\boldsymbol{y}^*$, of the dual polytope (by showing that the complementary slackness conditions are satisfied).

Finally, we show how to use $\boldsymbol{y}^*$ to define ψ in a specific manner so that $\Delta(\pi, \psi) \cdot \boldsymbol{y}^* \leq 0$ for each instance π (observe that this function ψ does not depend on π and hence it works for all instances). This function is precisely the function used in the algorithm. This ensures that the performance of our algorithm on each instance matches that of BALANCE on unit bid instances and is at least $1 - 1/e$.

We call this ensemble $L(\pi, \psi)$ a *tradeoff revealing family of LP's*. Once the competitive ratio of the algorithm for the unit bid case is determined via a factor-revealing LP, this family helps us find a tradeoff function that ensures the same competitive ratio for the arbitrary bids case.

1.3 Subsequent Developments

Over the last two years, since the conference version of this paper appeared in 2005 [MSVV05], the sponsored search market has been the subject of considerable study, both algorithmic and game theoretic. In what follows, we will give brief descriptions of some of the more related or significant works. For a more detailed exposition of these results, we refer the reader to [LPSV].

The online allocation problem: Buchbinder et al. [BJN] give a simple primal-dual algorithm and analysis for the adwords problem achieving the same competitive ratio as ours.

Mahdian et al. [MNS07] study the adwords problem when the search engine has a somewhat reliable estimate of the number of users searching for each keyword. They propose and analyze an algorithm that takes advantage of the given estimates of the frequencies of keywords to compute a near-optimal solution when the estimates are accurate, while at the same time maintaining a good worst-case competitive ratio in case the estimates are totally incorrect.

Goel and Mehta [GM07] analyze the performance of the greedy algorithm (which assigns each query to the highest bidder) in a distributional input model with queries arriving in a random permutation. They prove a tight competitive ratio of $1 - 1/e$.

Static models for ranking auctions: A large number of papers in this area study the auctions used by search engines for ranking the advertisements in a page. These models usually ignore the repeated nature of these auctions and focus on the equilibrium of a single auction.

[EOS05, Var06] investigate the equilibrium of generalized second-price auction (GSP), the charging scheme used by many search engines. Although GSP looks similar to the Vickrey-Clarke-Groves (VCG) mechanism, it generally does not have an equilibrium in dominant strategies, and truth-telling is not an equilibrium of GSP. [EOS05] describe the generalized English auction that corresponds to the GSP and show that it has a unique equilibrium, with the same payoffs to all players as the dominant strategy equilibrium of VCG.

The interested reader should also consult Crawford and Knoer [CK81] and Demange, Gale, and Sotomayor [DGS86] (which is a variant of the Hungarian algorithm for solving the assignment problem). Furthermore, the explicit form of incentive compatible payments for ranking auctions is carried out in [AGM06, IK06].

Click-fraud and cost-per-acquisition auctions: Another important issue in the context of online advertising is click-fraud — fraudulent clicks generated to deplete a competitors' budget.

Immorlica et al. [IJMT05] study this problem and present a click-fraud resistant method for learning the click-through rate of advertisements [IJMT05].

Another solution for addressing the above problem is to use a Cost-Per-Action or Cost-Per-Acquisition (CPA) charging scheme in which instead of paying for the click, the advertiser pays only when the user takes a specific action or completes a transaction. For a game theoretic analysis of these auctions see [NSV07, GP07].

Dispensing with auctions: In a different direction, [Vaz06] considers the scenario where keywords are sold at fixed prices rather than through auctions. They design a suitable utility function via which advertisers can express their preferences, and a polynomial time algorithm for computing equilibrium prices.

2 Problem Definition

The adwords problem is the following: There are N bidders, each with a specified daily budget b_i. Q is a set of query words. Each bidder i specifies a bid c_{iq} for query word $q \in Q$. A sequence $q_1 q_2 \ldots q_M$ of query words $q_j \in Q$ arrive online during the day, and each query q_j must be assigned to some bidder i (for a revenue of c_{iq_j}). The objective is to maximize the total revenue at the end of the day while respecting the daily budgets of the bidders.

Throughout this paper we will make the assumption that each bid is small compared to the corresponding budget, i.e., $\max_j c_{ij}$ is small compared to b_i, for all i. For the applications of this problem mentioned in the Introduction, this is a reasonable assumption.

An online algorithm is said to be α-competitive if for every instance, the ratio of the revenue of the online algorithm to the revenue of the best off-line algorithm is at least α.

While presenting the algorithm and the proofs, we will make the simplifying assumptions that the budgets of all bidders are equal (assumed unit) and that the best offline algorithm exhausts the budget of each bidder. These assumptions will be relaxed in Section 6.

3 A Discretized Version of the Algorithm

Let us first consider a greedy algorithm that maximizes revenue accrued at each step. It is easy to see that this algorithm achieves a competitive ratio of $\frac{1}{2}$ (see, e.g., [LLN01]); moreover, this is tight as shown by the following example with only two bidders and two query words: Suppose both bidders have unit budget. The two bidders bid c and $c+\epsilon$ respectively on query word q, and they bid 0 and c on query word q'. The query sequence consists of a number of occurrences of q followed by a number of occurrences of q'. The query words q are awarded to bidder 2, and are just enough in number to exhaust his budget. When query words q' arrive, bidder 2's budget is exhausted and bidder 1 is not interested in this query word, and they accrue no further revenue.

Our algorithm rectifies this situation by taking into consideration not only the bids but also the unspent budget of each bidder. For the analysis it is convenient to discretize the budgets as follows: we pick a large integer k, and discretize the budget of each bidder into k equal

parts (called *slabs*) numbered 1 through k. Each bidder spends money in slab j before moving to slab $j+1$.

Definition: At any time during the run of the algorithm, we will denote by slab(i) the currently active slab for bidder i.

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

$$\psi_k(i) = 1 - e^{-(1-i/k)}$$

Note that $\psi_k \rightarrow \psi$ as $k \rightarrow \infty$.

Discrete Version of the Algorithm

When a new query arrives, let the bid of bidder i be $c(i)$. Allocate the query to the bidder i who maximizes $c(i) \times \psi_k(\text{slab}(i))$.

Note that in the special case when all the bids are equal, our algorithm works in the same way as the BALANCE algorithm of [KP00], for any monotonically decreasing tradeoff function.

4 Analyzing BALANCE using a Factor-Revealing LP

In this section we analyze the performance of our algorithm in the special case when all bids are equal. This is exactly the algorithm BALANCE of [KP00]. We give a simpler analysis of this algorithm using the notion of a factor-revealing LP. This technique was implicit in [MRJW77, GK98, MMSV01] and was formalized and made explicit in [JMS02, JMM$^+$03]. We will see how to extend the analysis to the general case in Section 5. For another simple proof for BALANCE see [AL06].

We will assume for simplicity that in the optimum solution, each of the N players spends his entire budget, and thus the total revenue is N (the proof is similar even without this assumption, and we provide it in Section 6). Recall that BALANCE awards each query to the interested bidder who has the maximum unspent budget. We wish to lower bound the total revenue achieved by BALANCE. Let us define the *type* of a bidder according to the fraction of budget spent by that bidder at the end of the algorithm BALANCE: say that the bidder is of type j if the fraction of his budget spent at the end of the algorithm lies in the range $((j-1)/k, j/k]$. By convention a bidder who spends none of his budget is assigned type 1.

Clearly bidders of type j for small values of j contribute little to the total revenue. The factor revealing LP for the performance of the algorithm BALANCE will proceed by bounding the number of such bidders of type j.

Lemma 1 *If OPT assigns query q to a bidder B of type $j \leq k-1$, then BALANCE pays for q from some slab i such that $i \leq j$.*

The lemma follows immediately from the criterion used by BALANCE for assigning queries to bidders: B has type $j \leq k-1$ and therefore spends at most $j/k < 1$ fraction of his budget at the end of BALANCE. It follows that when query q arrives, B is available to BALANCE

for allocating q, and therefore B must allocate q to some bidder who has spent at most j/k fraction of his budget.

For simplicity we will assume that bidders of type i spend exactly i/k fraction of their budget, and that queries do not straddle slabs. The latter is justified by the fact that bids are small compared to budgets (e.g. taking bids to be smaller than $\frac{1}{k^2}$ of the budget). The total error resulting from this simplification is at most N/k and is negligible, once we take k to be large enough. Now, for $i = 1, 2, \ldots, k-1$, let x_i be the number of bidders of type(i). Let β_i denote the total money spent by the bidders from slab i in the run of BALANCE. It is easy to see (Figure 1) that $\beta_1 = N/k$, and for $2 \leq i \leq k$, $\beta_i = N/k - (x_1 + \ldots + x_{i-1})/k$.

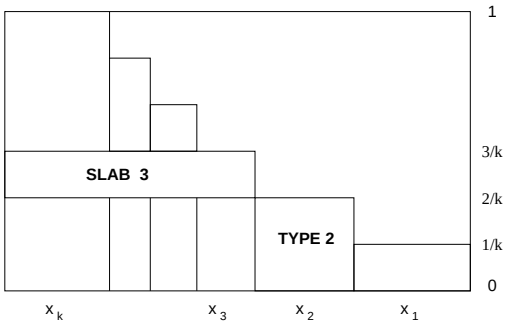

Figure 1: The bidders are ordered from right to left in order of increasing type. We have labeled here the bidders of type 2 and the money in slab 3.

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 : By Lemma 1,

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

The lemma follows by rearranging terms. □

The revenue of the algorithm is

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

To find a lower bound on the performance of BALANCE 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. In both the constraints below, i ranges from 1 to $k-1$.

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

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

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

Define $\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c}$ so the primal LP, L, can be written as

$$\max \quad \boldsymbol{c} \cdot \boldsymbol{x} \quad s.t. \quad \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b} \quad \boldsymbol{x} \geq 0.$$

and the dual LP, D, can be written as

$$\min \quad \boldsymbol{b} \cdot \boldsymbol{y} \quad s.t. \quad \boldsymbol{A}^T \boldsymbol{y} \geq \boldsymbol{c} \quad \boldsymbol{y} \geq 0.$$

Lemma 3 *As $k \to \infty$, the value Φ of the linear programs L and D goes to $\frac{N}{e}$*

Proof : On setting all the primal constraints to equality and solving the resulting system, we get a feasible solution $x_i^* \geq 0$. Similarly, we can set all the dual constraints to equality and solve the resulting system to get a feasible dual solution.

These two feasible solutions are:

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

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

Clearly they satisfy all complementary slackness conditions, hence they are also optimal solutions of the primal and dual programs.

This gives an optimal objective function value of

$$\begin{aligned}\Phi &= \boldsymbol{c}\cdot\boldsymbol{x}^* = \boldsymbol{b}\cdot\boldsymbol{y}^* \\ &= \sum_{i=1}^{k-1}(\tfrac{k-i}{k})\tfrac{N}{k}(1-\tfrac{1}{k})^{i-1} \\ &= N(1-\tfrac{1}{k})^k\end{aligned}$$

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 find an instance of the problem (e.g., the one provided in [KP00]) such that at the end of the algorithm all the inequalities of the primal are tight, hence the competitive ratio of BALANCE is exactly $1-\frac{1}{e}$.

5 A Tradeoff-Revealing Family of LPs for the Adwords Problem

To generalize the algorithms of [KP00] to arbitrary bids, it is instructive to examine the special case with bids restricted to $\{0,1,2\}$. One natural algorithm to try assigns each query to a highest bidder, using the previous heuristic to break ties (largest remaining budget). We provide an example in the Appendix to show that such an algorithm achieves a competitive ratios strictly smaller and bounded away from $1-1/e$.

In this section we show how one can derive the optimal trade-off function between the bid and the (fraction of) unspent budget. Observe that even if we knew the correct tradeoff function, extending the methods of the previous section is difficult. The problem with mimicking the factor-revealing LP is that now the tradeoff between bid and unspent budget is subtle and the basic Lemma 1 which allowed us to write the inequalities in the LP no longer holds.

Here is how we proceed instead: For every monotonically decreasing tradeoff function ψ and every instance π of the adwords problem and write a new LP $L(\pi,\psi)$ for our algorithm using tradeoff function ψ run on the instance π. Of course, once we specify the algorithm as well as the input instance, the actual allocations of queries to bidders is completely determined. In particular, the number α_i of bidders of type i is fixed. $L(\pi,\psi)$ is the seemingly trivial LP obtained by taking the left hand side of each inequality in the factor revealing LP L and substituting $x_i = \alpha_i$ to obtain the right hand side. Formally:

Recall the LP L from the previous section:

$$\max \quad \boldsymbol{c}\cdot\boldsymbol{x} \quad s.t. \quad \boldsymbol{A}\boldsymbol{x} \le \boldsymbol{b} \quad \boldsymbol{x} \ge 0$$

Let $\boldsymbol{a}$ be a $k-1$ dimensional vector whose ith component is α_i. Let $\boldsymbol{A}\boldsymbol{a} = \boldsymbol{l}$. We denote the following LP by $L(\pi,\psi)$:

$$\max \quad \boldsymbol{c}\cdot\boldsymbol{x} \quad s.t. \quad \boldsymbol{A}\boldsymbol{x} \le \boldsymbol{l} \quad \boldsymbol{x} \ge 0$$

The dual LP is denoted by $D(\pi, \psi)$ and is:

$$\min \quad \boldsymbol{l} \cdot \boldsymbol{y} \quad s.t. \quad \boldsymbol{A}^T \boldsymbol{y} \geq \boldsymbol{c} \quad \boldsymbol{y} \geq 0$$

Clearly, any one LP $L(\pi, \psi)$ offers no insight into the performance of our algorithm; after all the right hand sides of the inequalities are expressed in terms of the unknown number of bidders of type i. Nevertheless, the entire family $L(\pi, \psi)$ does contain useful information which is revealed by considering the duals of these LP's.

Since $L(\pi, \psi)$ differs from L only in the right hand side, the dual $D(\pi, \psi)$ differs from D only in the dual objective function; the constraints remain unchanged. Hence solution $\boldsymbol{y}^*$ of D is feasible for $D(\pi, \psi)$ as well. Recall that this solution was obtained by setting all nontrivial inequalities of D to equality.

Now by construction, if we set all the nontrivial inequalities of LP $L(\pi, \psi)$ to equality we get a feasible solution, namely $\boldsymbol{a}$. Clearly, $\boldsymbol{a}$ and $\boldsymbol{y}^*$ satisfy all complementary slackness conditions. Therefore they are both optimal. Hence we get:

Lemma 4 *For any instance π and monotonically decreasing tradeoff function ψ, $\boldsymbol{y}^*$ is an optimal solution to $D(\pi, \psi)$.*

The structure of the algorithm does constrain how the LP L differs from $L(\pi, \psi)$. This is what we will explore now.

As in the analysis of BALANCE, we divide the budget of each bidder into k equal *slabs*, numbered 1 to k. Money in slab i is spent before moving to slab $i+1$. We say that a bidder is of type j if the fraction of his budget spent at the end of the algorithm lies in the range $((j-1)/k, j/k]$. By convention a bidder who spends none of his budget is assigned type 1. As before, we make the simplifying assumption (at the cost of a negligible error term) that bidders of type j spend exactly j/k fraction of their budget. Let α_j denote the number of bidders of type j. Let β_i denote the total money spent by the bidders from slab i in the run of the algorithm. It is easy to see that $\beta_1 = N/k$, and for $2 \leq i \leq k$, $\beta_i = N/k - (\alpha_1 + \ldots + \alpha_{i-1})/k$.

Let $\Delta(\pi, \psi)$ be a $k-1$ dimensional vector whose ith component is $(\alpha_1 - \beta_1) + \ldots + (\alpha_i - \beta_i)$. The following lemma relates the right hand side of the LPs L and $L(\pi, \psi)$.

Lemma 5 $\boldsymbol{l} = \boldsymbol{b} + \Delta(\pi, \psi)$.

Proof : Consider the ith components of the three vectors. We need to prove:

$$\alpha_1(1 + \tfrac{i-1}{k}) + \alpha_2(1 + \tfrac{i-2}{k}) + \ldots + \alpha_i$$

$$= \tfrac{iN}{k} + (\alpha_1 - \beta_1) + \ldots + (\alpha_i - \beta_i).$$

This equation follows using the fact that $\beta_i = N/k - (\alpha_1 + \ldots + \alpha_{i-1})/k$. □

We are interested in comparing the performance of our algorithm (abbreviated as ALG) with the optimal algorithm OPT. The following definitions focus on some relevant parameters comparing how ALG and OPT treat a query q:

Definition: Let $\text{ALG}(q)$ ($\text{OPT}(q)$) denote the revenue earned by the algorithm (OPT) for query q. Say that a query q is of *type* i if OPT assigns it to a bidder of type i, and say that q lies in *slab* i if the algorithm pays for it from slab i.

Lemma 6 *For each query q such that $1 \leq \text{type}(q) \leq k-1$,*

$$\text{OPT}(q)\psi(\text{type}(q)) \leq \text{ALG}(q)\psi(\text{slab}(q)).$$

Proof : Consider the arrival of q during the run of the algorithm. Since $\text{type}(q) \leq k-1$, the bidder b to whom OPT assigned this query is still actively bidding from some slab $j \leq \text{type}(q)$ at this time. The inequality in the lemma follows from the criterion used by the algorithm to assign queries, together with the monotonicity of ψ.

□

Lemma 7 $$\sum_{i=1}^{k-1} \psi(i)(\alpha_i - \beta_i) \leq \frac{N}{k}.$$

Proof : We start by observing that for $1 \leq i \leq k-1$:

$$\sum_{q:\text{type}(q)=i} \text{OPT}(q) = \alpha_i$$

$$\sum_{q:\text{slab}(q)=i} \text{ALG}(q) = \beta_i$$

By Lemma 6

$$\sum_{q:\text{type}(q)\leq k-1} [\text{OPT}(q)\psi(\text{type}(q)) - \text{ALG}(q)\psi(\text{slab}(q))]$$

$$\leq \quad 0.$$

Next observe that

$$\sum_{q:\text{type}(q)\leq k-1} \text{OPT}(q)\psi(\text{type}(q))$$

$$= \quad \sum_{i=1}^{k-1} \sum_{q:\text{type}(q)=i} \text{OPT}(q)\psi(i)$$

$$= \quad \sum_{i=1}^{k-1} \psi(i)\alpha_i.$$

And

$$\sum_{q:\text{type}(q)\leq k-1} \text{ALG}(q)\psi(\text{slab}(q))$$

$$
\begin{aligned}
&\leq \sum_{q:\mathrm{slab}(q)\leq k} \mathrm{ALG}(q)\psi(\mathrm{slab}(q)) \\
&\leq \sum_{q:\mathrm{slab}(q)\leq k-1} \mathrm{ALG}(q)\psi(\mathrm{slab}(q)) \ + \ \frac{N}{k} \\
&= \sum_{i=1}^{k-1} \sum_{q:\mathrm{slab}(q)=i} \mathrm{ALG}(q)\psi(i) \ + \ \frac{N}{k} \\
&= \sum_{i=1}^{k-1} \psi(i)\beta_i \ + \ \frac{N}{k}.
\end{aligned}
$$

The lemma follows from these three inequalities.

□

The final step consists of choosing the correct tradeoff function ψ as a function of the dual optimal solution $\boldsymbol{y}^*$ itself, so that for every instance π, the value of the optimal solution to $L(\pi, \psi)$ is at most that of L.

Theorem 8 *For function ψ_k defined as*

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

the competitive ratio of the algorithm is $(1 - \frac{1}{e})$, as k tends to infinity.

Proof : By Lemma 4, the optimal solution to $L(\pi, \psi)$ and $D(\pi, \psi)$ has value $\boldsymbol{l} \cdot \boldsymbol{y}^*$. By Lemma 5 this equals $(\boldsymbol{b} + \Delta) \cdot \boldsymbol{y}^* \leq N/e + \Delta \cdot \boldsymbol{y}^*$ (since $\boldsymbol{b} \cdot \boldsymbol{y}^* \leq N/e$, from Section 4).

Now,

$$
\begin{aligned}
\Delta \cdot \boldsymbol{y}^* &= \sum_{i=1}^{k-1} \boldsymbol{y}_i^*((\alpha_1 - \beta_1) + \ldots + (\alpha_i - \beta_i) \\
&= \sum_{i=1}^{k-1} (\alpha_i - \beta_i)(y_i^* + \ldots + y_{k-1}^*) \\
&= \sum_{i=1}^{k-1} (\alpha_i - \beta_i)\psi(i) \\
&\leq \frac{N}{k},
\end{aligned}
$$

where the last equality follows from our choice of the function ψ, and the inequality follows from Lemma 7. As k tends to infinity, we get that the competitive ratio of our algorithm is $(1 - \frac{1}{e})$.

□

The above analysis helped us derive the correct tradeoff function ψ together with the competitive ratio. However, the proof of the competitive ratio of the algorithm is simpler, once we are given the correct ψ. We give a quick sketch the main steps of such a proof below:

From the definitions of the α and β variables, we have the following relations:

$$\forall i: \quad \beta_i = \frac{N - \sum_{j=1}^{i-1} \alpha_j}{k}$$

Lemma 7 gives us

$$\sum_i \psi(i)\alpha_i \ \leq \ \sum_i \psi(i)\beta_i$$

where the choice of ψ is:

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

Combining these relations we get:

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

But the left side of the inequality above is precisely the amount of money left unspent at the end of the algorithm. This establishes that the competitive ratio is $1 - 1/e$.

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 β_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 $\beta_i = \sum_j \beta_i^j$. Let α_i be the amount of money that the optimal allocation gets from the bins of final type i. Let $\alpha = \sum_i \alpha_i$, be the total amount of money obtained in the optimal allocation.

Now the relations used in the direct proof at the end of Section 5 become

$$\forall i: \quad \beta_i \geq \frac{\alpha - \sum_{j=1}^{i} \alpha_j}{k}$$

$$\sum_i \psi(i)\alpha_i \ \leq \ \sum_i \psi(i)\beta_i$$

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. Search engine companies charge a lower amount: the next highest bid. There are different ways of defining "next highest bid". We can extend our analysis for two of these definitions: the next highest bid is chosen from 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 Lower Bound for Randomized Algorithms

In [KVV90] a lower bound of $1 - 1/e$ was proved for the competitive ratio of any randomized online algorithm for the online bipartite matching problem. Also, [KP00] 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. By suitably adapting the example used in [KVV90], we show a lower bound of $1 - 1/e$ for online randomized algorithms for the b-matching problem,

even for large b. This also resolves an open question from [KP98].

Theorem 9 *No randomized online algorithm can have a competitive ratio better than* $1 - 1/e$ *for the b-matching problem, for large b.*

Proof : By Yao's Lemma [Yao77], it suffices to present a distribution over inputs such that any deterministic algorithm obtains at most $1 - 1/e$ of the optimal allocation on the average. 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, with $1/\epsilon$ 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 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

In practice, there is a lot of statistical information available about search queries. If the queries were selected from a fixed probability distribution, then the allocation problem becomes an

offline problem. Though this is NP-complete for large bids, in the realistic case where the bids are small compared to budgets, a $1-\epsilon$ approximation can be obtained by linear programming [BHJ+04]. In practice, the query distribution fluctuates over time, varying with time of day, special events, etc. Therefore it is desirable to have a very simple, time efficient online allocation scheme.

Let us start by giving a model for the query sequence that formalizes both their statistical nature as well as their unpredictable fluctuations. In this model, the queries are drawn from an arbitrary fixed distribution for a period of time. The distribution is switched by an adversary a number of times each day. Our goal is to design an online algorithm that achieves a $1-o(1)$ performance ratio against any fixed probability distribution, while achieving a good worst-case performance ratio when the distribution is switched suddenly (or evolves rapidly).

We believe a simple modification of our algorithm achieves this. Here is a concrete proposal: each bidder is assigned a weight, and his effective bid for a keyword is defined to be the product of the actual bid and his weight. We modify our algorithm to use effective bids in place of bids. The main open question here is whether for any fixed distribution on queries there is always a set of weights such that this algorithm achieves $1-o(1)$ expected competitive ratio.

How do we actually find a good set of weights for the bidders? Here is an online heuristic might provide a quick way of computing such weights: consider the allocation of queries for some window of time under the current weights. Adjust the weight of a bidder upwards if that bidder spends less than his fair share of his his budget during this time window, and downwards if he spends more than his fair share of the budget. Repeat this process after each such window of time.

In an earlier version of this paper [MSVV05], we had presented a second algorithm based on the RANKING algorithm for online bipartite matching of [KVV90]. The algorithm randomly permutes the bidders, and assigns each query to the bidder who maximizes the product of his bid for the query and the value of a particular function of his rank (position) in the permutation. The function used was the same as the function used in this paper for scaling the budgets. We claimed in [MSVV05] that this algorithm also achieves a competitive ratio of $1-1/e$ (for the expected revenue). There was a gap in our proof. What was actually proved in [MSVV05] is that a modification of this algorithm achieves factor $1-1/e$. This modification, which was called Refusal, is introduced just for the purposes of algorithm analysis and is not implementable, since it relies on knowledge of the optimal allocation. The gap in [MSVV05] was that unlike in [KVV90], the competitive ratio of Refusal is not necessarily a lower bound on that of the actual algorithm (which is what we claimed). Analyzing the competitive ratio for this algorithm remains an open question.

However, there is yet another way to generalize RANKING for general bids and large budgets. For a suitably large m, we can represent every bidder i with budget B_i by m bidders each with budget B_i/m and the same bid values. Now we can run the previous generalization of RANKING for new bidders. As the queries arrive and get allocated by the algorithm, the representatives of each bidder will run out of budget in the order in which they appear in the permutation. Since m is large, we can expect these representatives to be evenly distributed in the permutation. Therefore, the new extension of RANKING will roughly simulate the algorithm we explained in Section 3 and it should have the same competitive factor of $1-1/e$.

This line of argument indicates that the algorithm analyzed in this paper may be regarded as a common generalization of RANKING and BALANCED in the case of large budgets.

The use of our trade-off function for solving the adwords problem is reminiscent of the use of potential functions for online minimization problems, e.g., makespan minimization [AAF+97]. An interesting question is to see whether our methods can be used to derive the relevant trade-offs in the context of these problems. More generally, understanding the scope and nature of the main technique of our paper remains open.

Gaming by advertisers is a serious problem in online ad auctions. Our algorithm appears to provide some resilience against gaming schemes. One such scheme exploits the second-price auction to deplete the competitor's budget (unlike the Vickrey auctions, in this setting because of repeated play, second price auctions are not incentive compatible). This is done by bidding just short of the winning bid, thus quickly depleting the competitor's budget (this can be accomplished by a ghost bidder with small budget). Once the competitor is eliminated, the keyword can be obtained at a low bid. Our algorithm will often award query words to the ghost bidder thereby depleting his budget too. These are heuristic considerations. More formally, [BCI+05] gave some evidence that it is impossible to design a truthful mechanism in the presence of budget constraints. More generally , the uncertainty induced by the tradeoff in our algorithm seems to have the effect of increasing the competition in the auction.

Acknowledgments: We would like to thank Meredith Goldsmith, Kamal Jain, Subrahmanyam Kalyanasundaram, Milena Mihail, Sandeep Pandey, Serge Plotkin and Kunal Talwar for valuable discussions.

References

[AAF+97] J. Aspnes, Y. Azar, A. Fiat, S. Plotkin, and O. Waarts. On-line routing of virtual circuits with applications to load balancing and machine scheduling. *Journal of the ACM (JACM)*, 44(3):486–504, 1997.

[AGM06] Gagan Aggarwal, Ashish Goel, and Rajeev Motwani. Truthful auctions for pricing search keywords. In *Proceedings of the ACM Conference on Electronic Commerce (EC)*, 2006.

[AL06] Y. Azar and A. Litichevskey. Maximizing Throughput in Multi-Queue Switches. *Algorithmica*, 45(1):69–90, 2006.

[Bat05] John Battelle. *The search: how google and its rivals rewrote the rules of business and transformed our culture.* portfolio trade, 2005.

[BCI+05] Christian Borgs, Jennifer Chayes, Nicole Immorlica, Mohammad Mahdian, and Amin Saberi. Multi-unit auctions with budget-constrained bidders. In *ACM conference on Electronic Commerce*, 2005.

[BFK+04] 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.

[BHJ+04] V. Bahl, M. Hajiaghayi, K. Jain, V. Mirrokni, L. Qiu, and A. Saberi. Cell breathing in wireless lan: algorithms and evaluation. *Manuscript*, 2004.

[BJN] Niv Buchbinder, Kamal Jain, and Joseph (Seffi) Naor. Online primal-dual algorithms for maximizing ad auctions revenue. to appear in ESA 2007.

[CK81] Vincent P. Crawford and Elsie M. Knoerr. Job matching with heterogeneous firms and workers. *Econometrica*, 49(2):437–450, 1981.

[DGS86] G. Demange, D. Gale, and M. Sotomayor. Multi-Item Auctions. *The Journal of Political Economy*, 94(4):863–872, 1986.

[EOS05] Benjamin Edelman, Michael Ostrovsky, and Michael Schwarz. Internet advertising and the generalized second price auction: Selling billions of dollars worth of keywords. NBER working paper 11765, November 2005.

[GK98] M. Goemans and J. Kleinberg. An improved approximation algorithm for the minimum latency problem. *Mathematical Programming*, 82:111–124, 1998.

[GM07] Gagan Goel and Aranyak Mehta. Adwords market: An analysis of the greedy online algorithm in distributional models. Manuscript, 2007.

[GP07] Rica Gonen and Elan Pavlov. An incentive-compatible multi-armed bandit mechanism. In *Third Workshop on Sponsored Search Auctions*, 2007.

[Hen04] Monika Henzinger. Private communication. 2004.

[IJMT05] Nicole Immorlica, Kamal Jain, Mohammad Mahdian, and Kunal Talwar. Click fraud resistant methods for learning click-through rates. In *Lecture Notes In Computer Science*, pages 34–45, 2005.

[IK06] Garud Iyengar and Anuj Kumar. Optimal keyword auctions. In *Proceedings of the Second Workshop on Sponsored Search Auctions*, 2006.

[JMM+03] 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.

[JMS02] Kamal Jain, Mohammad Mahdian, and Amin Saberi. A new greedy approach for facility location problems. In *STOC*, pages 731–740, 2002.

[KP98] 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.

[KP00] Bala Kalyanasundaram and Kirk R. Pruhs. An optimal deterministic algorithm for online b -matching. *Theoretical Computer Science*, 233(1–2):319–325, 2000.

[KVV90] 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.

[LLN01] 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.

[LPSV] Sebastien Lahaie, David Pennock, Amin Saberi, and Rakesh Vohra. *Sponsored Search*. Book chapter to appear in Algorithmic Game Theory (edited by Nisan et al.).

[MMSV01] M. Mahdian, E. Markakis, A. Saberi, and V. Vazirani. A greedy facility location algorithm analyzed using dual fitting. *RANDOM-APPROX*, pages 127–137, 2001.

[MNS07] Mohammad Mahdian, Hamid Nazerzadeh, and Amin Saberi. Allocating online advertisement space with unreliable estimates. In *ACM Conference on Electronic Commerce*, 2007.

[MRJW77] 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.

[MSVV05] Aranyak Mehta, Amin Saberi, Umesh V. Vazirani, and Vijay V. Vazirani. Adwords and generalized on-line matching. In *Annual IEEE Symposium on Foundations of Computer Science*, 2005.

[NSV07] Hamid Nazerzadeh, Amin Saberi, and Rakesh Vohra. Mechanism design based on cost-per-acquisition and applications to online advertising. Preprint, 2007.

[Var06] Hal R. Varian. Position auctions. Working Paper, February 2006.

[Vaz06] Vijay V. Vazirani. Spending constraint utilities, with applications to the adwords market. submitted to *Math of Operations Research*, 2006

[Yao77] A. C. Yao. Probabilistic computations: towards a unified measure of complexity. *FOCS*, pages 222–227, 1977.

Appendix

A Counterexample for the Naive Algorithm

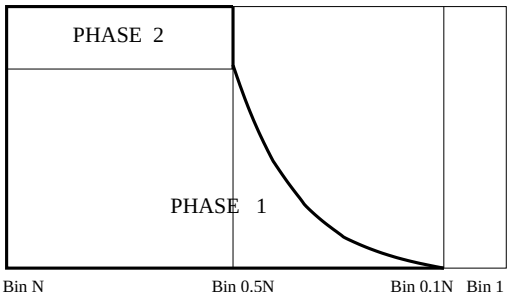

Figure 2: The bidders are ordered from right to left. The area inside the dark outline is the amount of money generated by the algorithm. The optimum allocation gets an amount equal to the whole rectangle.

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 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 2.

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 [KP00], and the factor is $1 - 1/e$.