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 optimal 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 [6]: 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 algorithm that awards each query to the highest bidder 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. In Section 9 we show that no randomized algorithm can achieve a better competitive ratio.

The offline version of the Adwords problem is **NP**-hard, and the best known approximation algorithm achieves a guarantee of $1-1/e$ [1], by solving a linear program followed by randomized rounding. Our online algorithms achieve the same approximation guarantee and are more efficient: the total running time is $O(NM)$ where N is the number of bidders and M the length of the query sequence.

[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.

In Section 7 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.

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. 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 [11], 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 [10] 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 results

To generalize the algorithms of [10] and [11] 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 algorithms are very simple, and are 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 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.

Algorithm 2:

Start by permuting the advertisers at random. Allocate the next query to the bidder maximizing the product of his bid and $\psi(r/n)$, where r 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.

1.3 A New Technique

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* [7]. 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 [10] result.

We give an LP, L, whose constraints are satisfied at the end of a run of BALANCE on any instance π 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., [7, 2], a factor-revealing LP is the only way known of deriving a tight analysis.

Next, we handle the case of arbitrary bids. We write a family of LP's $L(\pi, \psi)$, one for each input instance π and decreasing tradeoff function ψ. 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 $1 - 1/e$ on every instance π. Once the input instance and tradeoff functions are fixed, of course, the exact results achieved by the algorithm are completely determined. The right hand side of the inequalities in the LP $L(\pi, \psi)$ are based on these (unknown) parameters.

Now the constraints in each LP are obtained by relaxing a tautology, and therefore any single LP in this family does not contain any useful information. However, the entire family does express some of the structure of the problem which is revealed by considering the family of

dual linear programs $D(\pi, \psi)$.

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. Therefore, the dual programs $D(\pi, \psi)$ differ from D 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 the each LP in the family $D(\pi, \psi)$ attains its optimal value at the same vertex, $\boldsymbol{y}^*$, of the dual polytope. 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 Algorithm 1. This ensures that the performance of Algorithm 1 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.

The same proof outline also applies to Algorithm 2 once we suitably simplify the analysis of Karp, Vazirani and Vazirani [11] and cast it in terms of linear constraints.

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, and each query q_j must be assigned to some bidder i (for a bid of $c_{ij} = c_{iq_j}$). The objective is to maximize the total revenue while respecting the daily budgets of the bidders.

Throughout this paper we will make the assumption that the bids are small compared to the budgets, i.e., $\max_{i,j} c_{ij}$ is small compared to $\min_i b_i$. 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 algorithms and 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 7.

3 A Deterministic 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., [12]); 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.

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

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

Algorithm 1

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

Note that in the special case when all the bids are equal, our algorithm works in the same way as the BALANCE algorithm of [10], for any monotonically decreasing tradeoff function.

4 Analyzing BALANCE using a Factor-Revealing LP

In this section we analyze the performance of Algorithm 1 in the special case when all bids are equal. This is exactly the algorithm BALANCE of [10]. We give a simpler analysis of this algorithm using the notion of a factor-revealing LP. This technique was implicit in [14, 5, 13] and was formalized and made explicit in [8, 7].

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. 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 of type j, then BALANCE pays for q from slab k such that $k \leq j$.*

The lemma follows immediately from the criterion used by BALANCE for assigning queries to bidders.

For simplicity we will assume that bidders of type i spend exactly i/k fraction of their budget. The total error resulting from this simplification is at most N/k and is negligible. 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 ALG. It is easy to see (Figure 1) that $\beta_1 = N/k$, and for $2 \leq i \leq k$, $\beta_i = N/k - (x_1 + \ldots + x_{i-1})/k$.

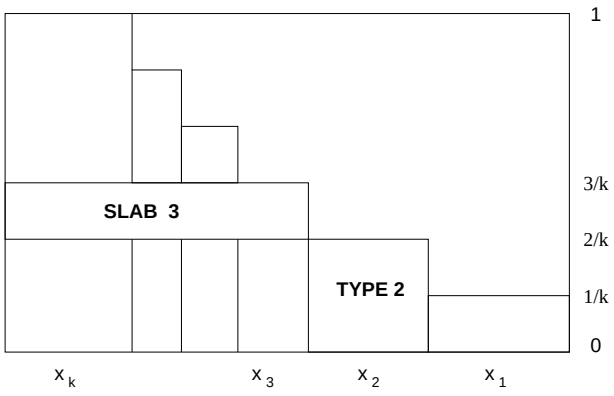

Figure 1: The bidders are ordered from right to left in order of increasing type. We have labeled here the bidders of type 2 and the money in slab 3.

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

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

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:

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

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

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

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

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

and the dual LP, D, can be written as

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

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

Proof : On setting all the primal constraints to equality and solving the resulting system, we get a feasible solution $x_i^* \ge 0$. 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:

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

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

But these clearly satisfy the 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 [10]) 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 A Tradeoff-Revealing Family of LPs for the Adwords Problem

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 for constant bids 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: We fix both a monotonically decreasing tradeoff function ψ as well as the instance π of the adwords problem and write a new LP $L(\pi,\psi)$ for Algorithm 1 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 and substituting $x_i = \alpha_i$ to obtain the right hand side. Formally:

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

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

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

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

Clearly, any one LP $L(\pi,\psi)$ offers no insight into the performance of Algorithm 1; 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 Algorithm 1 does constraint ow 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 Algorithm 1 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 Algorithm 1. It is easy to see that $\beta_1 = N/k$, and for $2 \leq i \leq k$, $\beta_i = N/k - (\alpha_1 + \ldots + \alpha_{i-1})/k$.

We are interested in comparing the performance of Algorithm 1 (abbreviated as ALG) with the optimal algorithm OPT. The following definitions focus on some relevant paramters comparing how ALG and OPT treat a query q:

Definition: Let $\text{ALG}(q)$ ($\text{OPT}(q)$) denote the revenue earned by Algorithm 1 (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 Algorithm 1 pays for it from slab i.

Lemma 5 *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 Algorithm 1. 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 Algorithm 1 to assign querries, together with the monotonicity of ψ.

□

Lemma 6 $$\sum_{i=1}^{k-1} \psi(i)(\alpha_i - \beta_i) \leq 0.$$

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 5

$$\sum_{q:\text{type}(q)\leq k-1} [\text{OPT}(q)\psi(\text{type}(q)) - \text{ALG}(q)\psi(\text{slab}(q))] \quad \leq \quad 0.$$

Next observe that

$$\begin{aligned}\sum_{q:\text{type}(q)\leq k-1} \text{OPT}(q)\psi(\text{type}(q)) = & \sum_{i=1}^{k-1} \sum_{q:\text{type}(q)=i} \text{OPT}(q)\psi(i) \\ = & \sum_{i=1}^{k-1} \psi(i)\alpha_i.\end{aligned}$$

And

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

The lemma follows from these three inequalities. □

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 7 $\quad \boldsymbol{l} = \boldsymbol{b} + \Delta(\pi, \psi).$

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 $\beta_i = N/k - (\alpha_1 + \ldots + \alpha_{i-1})/k$. □

Theorem 8 *For function ψ 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 4, the optimal solution to $L(\pi, \psi)$ and $D(\pi, \psi)$ has value $\boldsymbol{l} \cdot \boldsymbol{y}^*$. By Lemma 7 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,

$$\begin{aligned}
\Delta \cdot \boldsymbol{y}^* &= \sum_{i=1}^{k-1} \boldsymbol{y}_i^* ((\alpha_1 - \beta_1) + \ldots + (\alpha_i - \beta_i) \\
&= \sum_{i=1}^{k-1} (\alpha_i - \beta_i)(y_i^* + \ldots + y_{k-1}^*) \\
&= \sum_{i=1}^{k-1} (\alpha_i - \beta_i)\psi(i) \\
&\leq 0,
\end{aligned}$$

where the last equality follows from our choice of the function ψ, and the inequality follows from Lemma 6. Hence the competitive ratio of Algorithm 1 is $(1 - \frac{1}{e})$.

□

6 Direct Analysis of Algorithm 1

In the last section we derived the correct tradeoff function ψ to use in Algorithm 1, and in the process also gave a proof that the competitive ratio of the resulting algorithm is $1 - 1/e$. In this section abstract out the essential features of the argument and sketch a direct proof of the competitive ratio starting with this tradeoff function ψ.

Theorem 9 *The competitive ratio of Algorithm 1 is* $1 - 1/e$.

Proof : Recall that α_i is the number of bidders of type i, and β_i is the total amount of money spent by bidders from slab i. We have the following relations from Section 5:

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

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

Using the above equations and the choice of ψ:

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

we get:

$$\sum_{i=1}^{k} \alpha_i \frac{k-i+1}{k} \leq \frac{N}{e} \tag{2}$$

But the left side of (2) 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 $1 - 1/e$. □

7 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 Section 6 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 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.

8 A Randomized Algorithm

In this section we define a generalization of the RANKING algorithm of [11], 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 $b(i) \times \psi(\sigma(i))$.

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 6.

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.

Let b_{max} be the maximum bid value for any query from any bidder.

Lemma 10 *For any bidder, the amount of money which is not disallocated and not spent is at most b_{max}.*

Proof : Consider any bidder i and consider the queries that the optimum allcation allocates to i. The sum of the money spent by i in the optimal allocation is exactly the budget of i by assumption. For each such query q, let opt_q be the revenue of OPT on query q. When q enters during the algorithm, either it is allocated to i at the price of opt_q, or it is allocated to some other bidder. In the latter case, if i had opt_q amount of money remaining at that time, then opt_q amount is disallocated from i. Otherwise, the money remaining with i is less than $opt_q \leq b_{max}$. □

Lemma 11 *The competitive ratio of Refusal is at most the competitive ratio of Algorithm 2.*

Proof : By induction on the arrival of queries it is easy to see that the amount of money left with each bidder in Refusal is at most the amount of money with that bidder in 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 12

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

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 13

$$\forall i: \quad w_i \;\geq\; 1 - \frac{\sum_{j=1}^{i} x_j}{n} - b_{max} \tag{4}$$

Proof : By Lemma 10, 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 6, we see that the constraints (3) and (4) are similar to the constraints obtained in that proof. The amount of money left unspent also has the same form, namely

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

Hence we get

Proposition 14 *The competitive ratio of Refusal is at least* $1 - 1/e$.

From Lemma 11, we get:

Theorem 15 *The competitive ratio of the Algorithm 2 is at least* $1 - 1/e$.

Remark: Lemma 10 points to the reason why we assume that the largest bid is small compared to the budgets. Our analysis loses an amount of b_{max} due to fence-post errors, and it would be interesting to tighten the analysis and remove the assumption that the budget is large compared to the bids.

9 A Lower Bound for Randomized Algorithms

In [11], 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, [10] 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 [9].

Theorem 16 *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 [15], 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. □

10 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 concrete way of accomplishing this is by applying the algorithm for learning from expert advice [4] to fine tuning the tradeoff function. For example, the tradeoff function should ideally be relaxed towards the end of the day. This can be accomplished by consider a set of "experts" each of whom changes the tradeoff function according to a fixed schedule over the course of the day. The learning algorithm is used over successive days to track the best such daily variation in tradeoff function.

Our 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.

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 [11] 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] Y. Freund and R. Schapire. Adaptive game playing using multiplicative weights. *Games and Economic Behavior*, 29:79–103, 1999.

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

[6] Monika Henzinger. Private communication. 2004.

[7] 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.

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

[9] 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.

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

[11] 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.

[12] 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.

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

[14] 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.

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

Appendix

A Counterexample 1

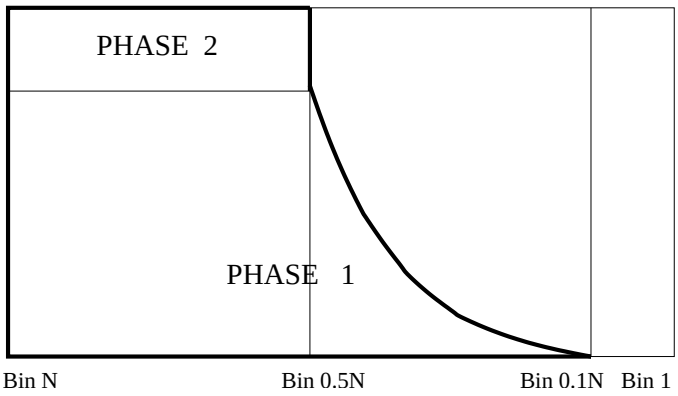

Figure 2: The bidders are ordered from right to left. The area inside the dark outline is the amount of money generated by the algorithm. The optimum allocation gets an amount equal to the whole rectangle.

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

B Counterexample 2

The example consists of an $N \times N$ upper-triangular matrix, as shown in Figure 3. The columns represent the queries and are ordered from right to left. The rows represent the bidders. The

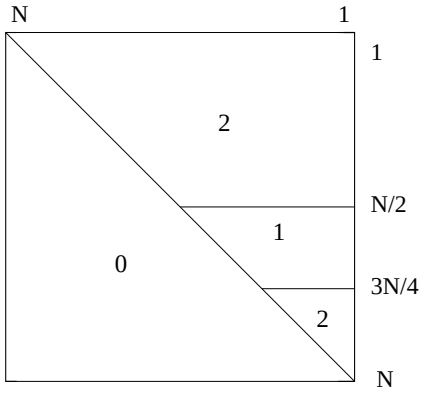

Figure 3: The rows N/2 to 3N/4 have a budget of 1. All other rows have a budget of 2.

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.