

Online Budgeted Matching in Random Input Models with applications to Adwords

Gagan Goel* Aranyak Mehta†

Abstract

We study an online assignment problem, motivated by *Adwords Allocation*, in which queries are to be assigned to bidders with budget constraints. We analyze the performance of the Greedy algorithm (which assigns each query to the highest bidder) in a randomized input model with queries arriving in a random permutation. Our main result is a tight analysis of Greedy in this model showing that it has a competitive ratio of $1 - 1/e$ for maximizing the value of the assignment. We also consider the more standard *i.i.d.* model of input, and show that our analysis holds there as well. This is to be contrasted with the worst case analysis of [MSVV05] which shows that Greedy has a ratio of $1/2$, and that the optimal algorithm presented there has a ratio of $1 - 1/e$. The analysis of Greedy is important in the Adwords setting because it is the natural allocation algorithm for an auction-style process.

From a theoretical perspective, our result simplifies and generalizes the classic algorithm of Karp, Vazirani and Vazirani for online bipartite matching. Our results include a new proof to show that the Ranking algorithm of [KVV90] has a ratio of $1 - 1/e$ in the worst case. It has been recently discovered [KV07] (independent of our results) that one of the crucial lemmas in [KVV90], related to a certain reduction, is incorrect. Our proof is direct, in that it does not go via such a reduction, which also enables us to generalize the analysis to our online assignment problem.

1 Introduction

In this paper we study an online assignment problem in which there is a random stream of items (or *queries*) arriving online, and each item, as it arrives, has to be allocated to one of several bins (or *bidders*). Each bin has a different value for the item, and also has a capacity (or *budget*). The goal is to maximize the total value of the allocation.

The motivation for this problem is keyword based ad auctions. In such an auction, advertisers report to the search engine their bids for different keywords (how much they are willing to pay for each click on their ad if it is placed next to the search results for that keyword). Besides submitting bids for different keywords, an advertiser also reports a global daily budget, over all keywords. When a user submits a query (consisting of keywords), she is returned, next to the search results, a small list of advertisements relevant to the keywords. The search engine's job is to decide which ads to return with each query in the query sequence.

The Adwords auctions, being crucial for the multi-billion dollar markets they support, call for theoretical investigation and there are several different issues that have been studied (e.g., the study of game theoretic aspects of single query auctions [AGM06, EOS07, Var06] and optimizations from advertisers' perspective [RW06, FMPS07]). Arguably, one major issue is that of designing allocation algorithms to maximize revenue (see [MSVV05, LPSV07]). An allocation algorithm has to determine for each query, as it arrives, which bidders win the query, i.e., whose ads should be displayed. A natural algorithm for such an auction style process is *Greedy*: Select the highest bidders for each query. It can be shown that the Greedy algorithm has a competitive ratio of $1/2$ in the standard online worst-case competitive analysis model. In [MSVV05] a deterministic algorithm with a competitive ratio of $1 - 1/e \simeq 0.63$ was presented, which was proved to be optimal, even over randomized algorithms¹.

*College of Computing, Georgia Institute of Technology, Atlanta GA, gagang@cc.gatech.edu

†Google, Inc., Mountain View, CA, aranyak@google.com (Work done while the author was at the IBM Almaden Research Center, San Jose, CA.)

¹In search auctions, winners are usually charged the next highest bid. Furthermore, each bid can be normalized by quality factors.

In spite of the elegance and optimal competitive ratio of this algorithm, there remains some dissatisfaction with the use of worst case analysis in that we are unlikely to see a worst-case sequence of queries as input. In this paper we make a more distributional assumption about the query sequence: the queries arrive in a random permutation, i.e. the *set* of queries is still arbitrary, but the *order* of queries is random. This leads us to formulate the following combinatorial problem (we assume, for simplicity, that there is exactly one ad returned per query):

There are m bidders. Bidder i has a budget of B_i . There is a (unknown) set Q of n queries. For $q \in Q$, and $i \in [m]$, bid_{iq} is the bid of bidder i for query q . The queries arrive online one at a time, in a random permutation of Q . The algorithm has to allocate each query, as it arrives, to a bidder. Let Q_i be the set of queries allocated to bidder i at the end of the input sequence. The revenue of the algorithm is defined as $\sum_{i=1}^m \min\{B_i, \sum_{q \in Q_i} bid_{iq}\}$. The goal of the algorithm is to maximize its revenue.

We also study this assignment problem in the more standard *i.i.d.* input model, in which there is an unknown distribution on queries, and the next query in the sequence is picked independently from this distribution.

Understanding the Adwords allocation problem in distributional models is important, and was an open question in [MSVV05] and in [LPSV07] (see also a different approach in [MNS07]). The random permutations model has been studied, e.g., in the classic secretary problems [Dyn63], recent work in auctions [BIK07] and in data streams [GM06] with the motivation that an adversary may have control over the set of inputs, but not have control over the order of the input, which may follow some random process.

In the problem above, we will assume that for each bidder $i \in m$: $\max_{q \in Q} bid_{iq}$ is very small compared to B_i . This is a reasonable assumption for the Adwords motivation, also made in [MSVV05]. But in fact, our algorithm and analysis work for a larger class, encompassing this assumption. We define this class formally in Section 3, but mention here that this class also contains the problems of bipartite matching (all bids 0 or 1, all budgets 1) and b -matching, for any positive integer b (all bids 0 or 1, all budgets b). The *offline* version of this problem is NP-hard, with the best known algorithm [AM04] (see also [FGMS06]) giving a factor of $1 - 1/e$, even if the bids are not restricted to be small compared to budgets (when the bids are very small compared to budgets then LP rounding gives a factor very close to 1). Finally, we note that independent of the Adwords motivation, the problem defined above is a general combinatorial allocation problem, closely related to classic matching problems, and with possible applications beyond the current motivation.

1.1 Main Results Our main result is an analysis to show that Greedy has competitive ratio $1 - 1/e$ in the random permutation input model, as well as in the *i.i.d.* model. This is to be compared with the ratios in the worst case input model: $1/2$ for Greedy and the optimal $1 - 1/e$ for the algorithm in [MSVV05]. We also show that our analysis is tight by providing examples in the two models for which Greedy has ratio exactly $1 - 1/e$. We also show a lower bound that no deterministic algorithm can have a ratio better than $3/4$ and no randomized algorithm can have a ratio better than $5/6$ for the case of bipartite matching in the random permutation model.

Along the way, we provide a direct proof for the RANKING algorithm of [KVV90] for the online bipartite matching problem (see below for details).

1.2 Techniques From a technical perspective our work can be seen as simplifying and generalizing the classic work of Karp, Vazirani and Vazirani on online bipartite matching (matching boys to girls) in the worst case input model [KVV90]. Their RANKING algorithm fixes beforehand a random permutation of the boys, and matches each arriving girl to the highest available neighbor in the permutation. They show a duality result that the performance of RANKING in the worst case is the same as the performance of Greedy in the random permutations model (boys arrive in a random permutation). They show that this competitive ratio is $1 - 1/e$, by analyzing a “virtual algorithm” called EARLY, which works identical to RANKING, but additionally, refuses to match a boy if the “correct” girl has already been matched off. They show that RANKING is strictly better than EARLY, and that EARLY has a competitive ratio of $1 - 1/e$.

For simplicity of presentation, we assume in our analysis that the bids are already normalized if necessary, and that we display only one ad per query and hence need to determine a single winner per query. We do not consider the issue of second price in our setting, and assume that a winning bidder pays his own bid. We also make a natural modeling assumption (also made in [MSVV05]) that all the bids of a bidder are very small compared to his budget.

On the way to proving our main result on the general Adwords problem, we provide a different proof for bipartite matching. This proof has the advantage that it is simpler and direct, in that we do not consider any “Refusal” algorithm (like EARLY) at all, but prove the factor of Greedy via direct arguments. This simpler analysis enables us to extend the scope of the algorithm and the analysis to our general assignment problem in the randomized input model. In fact, independent of our result, it has been discovered [KV07] that the proof in [KVV90] has a hole, in particular that one of the lemmas (Lemma 6) related to the EARLY algorithm is incorrect. Thus, while RANKING indeed has a ratio of $1 - 1/e$ (by our proof), the ratio of EARLY is unknown at this point.

Our direct proof for the case of matching follows the main ideas from [KVV90], but also introduces some new ideas. To provide a direct proof for Greedy we use a different classification of permutations: For each boy p , we classify the set of all permutations into those in which p remains unmatched and those in which p gets matched. We further classify the latter class into two subclasses – *good* and *bad* – depending on the structure of the match. We then bound the number of unmatched boys by showing that for every unmatched boy there is a certain fraction of bad matches, and for every bad match there is a certain fraction of good matches. This gives us the factor of Greedy.

The Tagging Procedure: In going from the case of matching to the more general case of Adwords, we need to develop some new techniques to take care of the complexity introduced in having different bids for the same query. The main reason is that in matching, since all the bids are 0 or 1 (with a budget of 1), any match is as good as another, while this is not the case with different bids. This creates several different (but related) issues: In matching, if a boy gets matched to a *wrong* girl then it can affect at most one other boy, in the sense that at most one other boy might not get matched. In the general problem, if a query goes to a *wrong* bidder then it can affect several other queries in many different ways. To capture this interaction we introduce a ‘*tagging procedure*’ which finds an appropriate map from a query to other queries. This is required in order to define notions of good and bad events. Another difference from matching is that a query may be tagged simultaneously by any number of good and bad tags, while in matching the mapping is always either good or bad, from exactly one boy to another.

On Monotonicity: One important property that we need in our analysis is that of *monotonicity* of the allocation algorithm: If you move a query to the front of the permutation, then each bidder spends at least as much money now as before (see Lemmas 2.5 and 3.1 for details). This property seems very useful to get a handle on the analysis: Our proof in the matching case exploits the fact that Greedy is monotonic, while the proof in [KVV90] has a hole precisely because their EARLY algorithm is not monotonic. However, upon moving to the general Adwords problem, it turns out that Greedy is *not monotonic* as such. This creates an obstacle in the proof. We get around it by analyzing a *variant of Greedy*: In the case that a bidder has not yet exhausted his budget, but his bid for a query is larger than his remaining budget, we do not truncate his bid, but instead over-spend his budget. Each bidder will overspend at most once, and we will not count the overspending as part of the revenue. This variant of Greedy can be shown to be monotonic, and therefore easier to analyze. We then show that the revenue of Greedy is almost the same as that of the variant, thus proving a bound for Greedy.

Regarding our random permutation model of input, we note that it is not new, used most notably in the classic secretary-style problems [Dyn63] (see also [BIK07]), in which a random permutation of a set of numbers arrives online and we have to pick the largest number when it arrives. Our problem is conceptually different, since it involves *partitioning* the entire random sequence into m parts to maximize the sum of the values of the parts, rather than picking one element or structure in the sequence which is the largest. Likewise, the techniques from secretary-style problems do not seem to apply here. The random permutations model has also been used recently in the data-streams literature [GM06].

We structure the rest of this paper as follows: We first study the simpler case of bipartite matching in Section 2, before analyzing the general problem in Section 3 (the proof in the case of matching is simpler and helps understand the general case). We analyze the i.i.d. model in Section 4.2, and provide the lower bounds in the permutation model in Section 4.3. We conclude with a general discussion of this and related problems in Section 5.

2 The Case of Online Bipartite Matching

In this section we consider the special case of bipartite matching (bids all 0 or 1, budgets all 1). In this problem, there is a bipartite graph (boys and girls). In the random permutations model, only the girls are known to the algorithm in advance, while the boys arrive in a random permutation. As a boy arrives, his incident edges are revealed, and he can be matched off to some girl. The Greedy algorithm matches each arriving boy to the first neighboring girl who is still available (first in some pre-determined arbitrary order). We proceed to analyze the performance of Greedy in this model. By a duality principle shown in [KVV90], this also gives a proof for RANKING in the online worst case model.

We label the boys and girls with numbers from 1 to n . We will use $p, q \in [n]$ to denote boys and girls, and use $s, t \in [n]$ to denote the positions in the permutation. We also use the notion of *time*: time t will denote the event when Greedy tries to allocate the boy at position t . We assume that there is an optimal matching of size n , and that $\forall p \in [n]$ it matches boy p with the girl p (this assumption is for simplicity of presentation, and can be removed later). Let Ω be the set of all permutations of $[n]$. For a permutation $\sigma \in \Omega$, $\sigma(s)$ will represent the boy at position s , and $\sigma^{-1}(p)$, the position of boy p .

2.1 Intuition: Whether a boy p gets matched or not clearly depends on his position in the random permutation. If he is high enough in the permutation, then he will get matched, but if he comes in low in the permutation, then by the time he arrives, his desirable girls may all have been matched off to other boys.

Let us say that boy p remained unmatched in a particular permutation σ . Note that this could only have happened if girl p was already matched (say, to boy p') when boy p arrived. So we can associate the *miss* of boy p to the *match* of boy p' . Likewise, we can associate every miss to a unique match. But notice that all these *matches* that we considered are of special kind: girl p got matched to boy p' and boy p arrived *after* boy p' . We will later define these kind matches to be *bad matches*. Now let us consider the second kind of *match*: i.e. girl p got matched to boy p' and boy p arrived *before* boy p' . We shall later define such a match of boy p' to be a *good match*. Note that in a good match we are guaranteed that both boy p and girl p got matched (as well as boy p'). Our goal is to prove that, on the average over all permutations, there are many matches, in particular, many good matches. Clearly, every miss for boy p is caused by a bad match for some other boy q (who stole his girl p). This immediately shows a factor of 1/2 for Greedy (in fact for every permutation). We will show something stronger: On average over all permutations, the misses result in an even larger fraction of bad and good matches.

2.2 Proof Outline: For each $\sigma \in \Omega$, $p \in [n]$ and $s \in [n]$, define:

$$miss_{\sigma}(s, p) = \begin{cases} 1 & \text{if } \sigma(s) = p \text{ and boy } p \text{ remains} \\ & \text{unmatched at the end of Greedy.} \\ 0 & \text{otherwise.} \end{cases}$$

$$good_{\sigma}(s, q) = \begin{cases} 1 & \text{if girl } q \text{ is matched to boy } \sigma(s), \\ & \text{and } \sigma^{-1}(\text{boy } q) \leq s. \\ 0 & \text{otherwise.} \end{cases}$$

$$bad_{\sigma}(s, q) = \begin{cases} 1 & \text{if girl } q \text{ is matched to boy } \sigma(s), \\ & \text{and } \sigma^{-1}(\text{boy } q) > s, \\ 0 & \text{otherwise.} \end{cases}$$

Define also the following *partitions* for every boy p . Partition the set Ω into groups of permutations s.t. in each group the relative order of all but boy p is fixed. Let Ω_p be one such group. Let $\sigma_k \in \Omega_p$ be the permutation in which boy p occurs at position k .

We will prove three main relationships between aggregates of these variables. The first inequality follows immediately from the definitions (either the boy in position t remains unmatched, or some girl gets matched to him). The other two require a detailed look at the relationship between different permutations, and their proofs

are provided immediately below in Section 2.3. Lemmas 2.2 and 2.3 hold for every $p \in [n]$ and every group Ω_p .

LEMMA 2.1.

$$\forall \sigma \in \Omega, \quad \forall t \in [n] : \\ \sum_p \text{good}_\sigma(t, p) + \sum_p \text{bad}_\sigma(t, p) + \sum_p \text{miss}_\sigma(t, p) = 1$$

LEMMA 2.2.

$$\forall t \in [n] : \quad \text{miss}_{\sigma_t}(t, p) \leq \sum_{\sigma \in \Omega_p} \sum_{s: s < t} \frac{\text{bad}_\sigma(s, p)}{n - s}$$

LEMMA 2.3.

$$\forall t \in [n - 1] : \\ \sum_{\sigma \in \Omega_p} \sum_{s \leq t} \text{bad}_\sigma(s, p) \frac{s}{n - s} \leq \sum_{\sigma \in \Omega_p} \sum_{s: s \leq t+1} \text{good}_\sigma(s, p)$$

The three lemmas above constrain the variables miss, bad and good at the end of Greedy, for every input. Note how Lemmas 2.2 and 2.3 show that if there are many misses then there are many bad matches and therefore many good matches.

Also notice that the revenue of Greedy is given by:

$$\sum_s \left[E_{\sigma \in \Omega} \left[\sum_p \text{bad}_\sigma(s, p) \right] + E_{\sigma \in \Omega} \left[\sum_p \text{good}_\sigma(s, p) \right] \right]$$

We minimize the revenue over the constrained region given by the inequalities to get the worst performance of the algorithm (on a random permutation of the worst set Q). This program turns out to be linear, and we can solve the LP to show a factor of $1 - 1/e$. This technique is known as a *factor revealing LP* and the details are given in the Appendix A.

THEOREM 2.1. *The competitive ratio of Greedy in the random permutations input model (and hence of RANKING in the online model) is $1 - 1/e$.*

2.3 Some Basic Properties of Greedy and Proofs of the Inequalities: We start by describing three basic properties (Lemmas 2.4, 2.5 and 2.6) of the Greedy algorithm and the variables defined above. These will help up work towards our main lemmas (Lemmas 2.2 and 2.3). The first lemma follows easily from the definitions.

LEMMA 2.4. **[Prefix Property]** *For any two permutations σ_1 and σ_2 , and any $t \in [n]$, if we have $\forall s \leq t : \sigma_1^{-1}(s) = \sigma_2^{-1}(s)$, then the runs of Greedy on σ_1 and σ_2 are identical from time 1 to time t .*

The second lemma describes a crucial monotonicity property of the Greedy algorithm. Informally, it says that if you move a boy up front in the permutation, then the set of girls who used to be matched by time t are now all matched by time $t + 1$. Formally, for $s \in [n]$, define $G_s(t)$ to be the set of the girls matched during the time 1 to t by Greedy in the run on permutation σ_s .

LEMMA 2.5. **[Monotonicity]** $\forall s, m \in [n], s < m :$

- (1) $\forall t \in [1, s - 1] : G_m(t) = G_s(t)$
- (2) $\forall t \in [s - 1, m - 1] : G_m(t) \subseteq G_s(t + 1)$

Proof. The first part follows directly from Lemma 2.4. We will use induction on t to prove the second part. The base case is when $t = s - 1$, and follows easily from the first part itself. Suppose the lemma holds for all $t \leq k$,

where $s - 1 \leq k < m - 1$. We will show that the lemma holds for $t = k + 1$. By hypothesis, $G_m(k) \subseteq G_s(k + 1)$. Notice that the boy at position $k + 1$ in σ_m is same as boy at position $k + 2$ in σ_s . Let us denote this boy by b .

If b is unmatched in the run of Greedy on σ_m , then the induction step holds trivially. So, instead, assume that b is matched to girl g in the run of Greedy on σ_m . We shall show that $g \in G_s(k + 2)$. Suppose not. This means that girl g was available when Greedy was matching b in σ_s , but the boy b got matched to some other girl $g' \notin G_s(k + 1)$. By the induction hypothesis we have $g' \notin G_m(k)$. Therefore both g and g' were available at time $k + 1$ in the run on σ_m , and Greedy should have preferred g' over g in that run as well, a contradiction. Hence $g \in G_s(k + 2)$ and the lemma holds by induction.

From the definitions, we know that for any permutation σ , at most one out of the $2n$ variables, $\{good_\sigma(s, p), bad_\sigma(s, p)\}_{s \in [n]}$ can be 1:

$$(2.1) \quad \forall \sigma : \quad \sum_t (good_\sigma(t, p) + bad_\sigma(t, p)) \leq 1$$

The third property tells us which variable is 1, for each permutation $\sigma_x \in \Omega_p$:

LEMMA 2.6. [Partition] *Fix p , and a partition Ω_p . Consider the run of Greedy on the permutation σ_n . If girl p gets matched to the boy at position t , for some $t \in [n]$, then:*

$$(1) \quad \forall x \in [1, t] : \quad \sum_{s: s \leq t+1} good_{\sigma_x}(s, p) = 1$$

$$(2) \quad \forall x \in [t + 1, n] : \quad bad_{\sigma_x}(t, p) = 1$$

Proof. (2): By Lemma 2.4 (Prefix Property), we know that $\forall x \in [t + 1, n]$, in the run on σ_x , girl p is matched to the boy at position t , and therefore $bad_{\sigma_x}(t, p) = 1$.

(1): By Lemma 2.5 (Monotonicity), we know that $\forall x \in [1, t]$, in the run on σ_x , girl p is matched to some boy whose position lies in the range $[x, t + 1]$, and therefore $\sum_{s \in [x, t+1]} good_{\sigma_x}(s, p) = 1$.

Now we are ready to prove the main inequalities (Lemmas 2.2 and 2.3):

Proof of Lemma 2.2 : If $miss_{\sigma_t}(t, p)$ equals 0, then lemma holds trivially. So assume that $miss_{\sigma_t}(t, p) = 1$. This means that in σ_t , boy p (who is in position t) remains unmatched. This is possible only if girl p is matched to some boy whose position is some $t^* < t$. Now by Lemma 2.6, $\forall s \in [t^* + 1, n]$, $bad_{\sigma_s}(t^*, p) = 1$. This gives the following sequence:

$$(2.2) \quad \sum_{\sigma \in \Omega_p} \sum_{s: s < t} bad_\sigma(s, p) / (n - s)$$

$$\geq \sum_{\sigma \in \Omega_p} bad_\sigma(t^*, p) / (n - t^*) \quad (t^* < t)$$

$$\geq \sum_{s=t^*+1}^n bad_{\sigma_s}(t^*, p) / (n - t^*)$$

$$= 1 \quad (\text{Lemma 2.6})$$

$$= miss_{\sigma_t}(t, p)$$

□

Proof of Lemma 2.3 : We consider two different cases: whether girl p get matched or remains unmatched in the run on σ_n .

Case 1: Girl p remains unmatched in σ_n . Take any permutation $\sigma_x \in \Omega_p$, $x \in [n]$. If girl p is matched in σ_x then, by Lemma 2.4 (Prefix Property), it could only be to a boy in some position greater than or equal to x . Hence, $\forall t: \text{bad}_{\sigma_x}(t, p) = 0$, and the lemma holds trivially.

Case 2: Girl p is matched in σ_n . Suppose she is matched to the boy in position t^* . There are two subcases:

Case 2.1: $t < t^*$. Lemma 2.6 and Lemma 2.1 tell us which of the variables are 1. We see that for every $\sigma \in \Omega_p$, and every $s < t^*$, $\text{bad}_\sigma(s, p) = 0$. Hence in this case, the left hand side is 0, and the lemma follows trivially.

Case 2.2: $t \geq t^*$. Again using Lemma 2.6 and Lemma 2.1 we get the following:

$$\begin{aligned} \sum_{\sigma \in \Omega_p} \sum_{s \leq t} \text{bad}_\sigma(s, p) \frac{s}{n-s} &= \sum_{\sigma \in \Omega_p} \text{bad}_\sigma(t^*, p) \frac{t^*}{n-t^*} \\ &= t^* \\ &= \sum_{\sigma \in \Omega_p} \sum_{s: s \leq t^*+1} \text{good}_\sigma(s, p) \\ &= \sum_{\sigma \in \Omega_p} \sum_{s: s \leq t+1} \text{good}_\sigma(s, p) \end{aligned}$$

All the equalities follow from Lemma 2.6 and Equation (2.1). The first equality holds because $\forall s \neq t^*, \forall \sigma \in \Omega_p, \text{bad}_\sigma(s, p) = 0$. The second equality holds because $\text{bad}_\sigma(t^*, p) = 1$ for precisely the last $n - t^*$ of the $\sigma \in \Omega_p$. Similarly, the last two equalities hold because $\sum_{s: s \leq t^*+1} \text{good}_\sigma(s, p) = 1$ for precisely the first t^* of the σ in Ω_p , and the rest of the variables are 0. This proves the lemma in this case as well. \square

3 The General Problem

In this section we study the general assignment problem as defined in Section 1. As mentioned there, our analysis holds for a larger class of inputs which includes bipartite matching and the case of small bids. We describe this class below:

For each bidder i , consider an allocation of some subset of the queries to bidder i , say (q_1, q_2, \dots, q_k) , which straddles the budget in the following sense: $\sum_{j=1}^{k-1} \text{bid}_{ij} < B_i \leq \sum_{j=1}^k \text{bid}_{ij}$. For every such allocation we are guaranteed that the overspending is small compared to the budget, i.e., $\sum_{j=1}^k \text{bid}_{ij} - B_i \ll B_i$.

We will show that Greedy, which assigns every query that arrives to the highest bidder with a non zero remaining budget, has a competitive ratio of $(1 - 1/e)$ when the input arrives in a random permutation of the (unknown) query set (the case of i.i.d. inputs is considered in Section 4.2). Note that OPT is an offline optimum algorithm and will have same allocation for any such input permutation.

Our analysis of Greedy in the matching case involved a mapping between a boy p and the boy to whom girl p got matched. In the more general case of Adwords allocation with different bids this kind of association becomes non trivial. For every query p there is a corresponding bidder i_p to whom OPT allocated query p . But now Greedy could assign several queries with different bid values to this bidder and the mapping between query p and these queries is not straightforward. Our tagging procedure takes care of this difficulty:

For a permutation $\sigma \in \Omega$, $\sigma(s)$ will represent the query at position s , and $\sigma^{-1}(p)$, the position of query p .

The Tagging Procedure: Fix an optimal allocation OPT and also fix, for each bidder, an arbitrary *ordering* of the queries assigned to it in OPT . Now consider any query q and the bidder i to whom OPT assigned q . In the fixed ordering, when OPT assigned q to i , suppose that the money spent by i increased from x to y . Then we will call $[x, y]_i$, the *Opt-Interval* of the query q : i.e., define $\text{Opt-Interval}(q) = [x, y]_i$. Greedy, run on some permutation σ , makes its own allocation, leading to a similar *Alg-Interval* $_\sigma(q)$ for each query q .

For a query q , the money in $\text{Opt-Interval}(q)$ is spent by Greedy on some other queries, say, e.g., q_1, q_2, q_3 . We say that q_1, q_2 and q_3 *tag* the *Opt-Interval* of q . We define $\text{tagset}_\sigma(q_1, q)$ as the set $\text{Opt-Interval}(q) \cap \text{Alg-Interval}_\sigma(q_1)$. The same set is also called $\text{goodset}_\sigma(q_1, q)$ or $\text{badset}_\sigma(q_1, q)$, depending on whether $\sigma^{-1}(q) \leq \sigma^{-1}(q_1)$ or $\sigma^{-1}(q) > \sigma^{-1}(q_1)$.

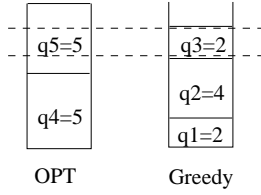


Figure 1: How bidder i 's money is spent in OPT and in the run of $Greedy$ on the permutation $(q_1, q_2, q_4, q_5, q_3)$. The dotted lines highlight $goodset_\sigma(q_3, q_5) = tagset_\sigma(q_3, q_5)$

Example: For illustration purposes, consider the following scenario. Focus on five special queries q_1, q_2, q_3, q_4, q_5 , and a bidder i who has a budget of 10. Fix an OPT which assigns q_4 and q_5 to bidder i in that order (see Fig 1). Consider a permutation σ in which the relative positions of these queries is $(q_1, q_2, q_4, q_5, q_3)$, and suppose that $Greedy$ assigns q_1, q_2, q_3 to bidder i (in that order). The bids of the different queries are shown in the figure. Then we have the following tagsets: $tagset_\sigma(q_1, q_4) = [0, 2]_i$, $tagset_\sigma(q_2, q_4) = [2, 5]_i$, $tagset_\sigma(q_1, q_5) = \emptyset$, $tagset_\sigma(q_2, q_5) = [5, 6]_i$, $tagset_\sigma(q_3, q_4) = \emptyset$, $tagset_\sigma(q_3, q_5) = [6, 8]_i$. The first four are *badsets* and the last two are *goodsets* (since q_3 is after q_4 and q_5 in the permutation). Note also that the subinterval $[8, 10]_i$ of $Opt-Interval(q_5)$ is not tagged. This contributes towards the loss of $Greedy$, and we will define it later as a miss of q_5 .

Note how a query's $Opt-Interval$ can be tagged simultaneously by any number of other queries, and simultaneously with good and bad tags (unlike in the case of matching, in which the tags were one-to-one). Also, if a bidder spends more money in $Greedy$ compared to in OPT then some part of $Alg-Interval_\sigma(q)$ may not even be used to tag any $Opt-Interval$. Define $extra_\sigma(q) = Alg-Interval_\sigma(q) \setminus \cup_p Opt-Interval(p)$.

Define $|[x, y]_i| = y - x$. Also, for any query q , we define $opt(q) = |Opt-Interval(q)|$ and $alg_\sigma(q) = |Alg-Interval_\sigma(q)|$. By convention, $|\emptyset| = 0$.

3.1 The Main Properties of Greedy and the Analysis Recall from Section 2 that in the case of matching, $Greedy$ has three properties (namely **Prefix**, **Monotonicity** and **Partition**) which we exploited in proving the relationships between the variables *miss*, *bad* and *good*. In the general case, as we saw above, $Greedy$ behaves in a much less controlled manner because of the different bids. However, we manage to prove that the corresponding properties still hold, in spite of this different behavior². This is sufficient to prove our main lemmas (Lemmas 3.4, 3.5 and 3.6), and prove a factor of $1 - 1/e$. We prove the three properties in this Section, and defer the proof of Lemmas 3.4, 3.5 and 3.6 to the appendix B.

Let us first explicitly spell out a very basic property of $Greedy$ (which we took for granted in matching):

Consistent Tie-Breaking: Consider a query q , and any two permutations σ, σ' . Define $highest_\sigma(q)$ as the set of bidders who have the highest bid for q among all bidders with non-zero remaining budget, when q arrives during the run on σ . Define $highest_{\sigma'}(q)$ similarly. Suppose that $highest_{\sigma'}(q) \subseteq highest_\sigma(q)$, and suppose that in the run on σ , q is allocated to bidder $i^* \in highest_{\sigma'}(q)$. Then it has to be true that q is allocated to i^* , even in the run on σ' .

We now proceed to prove the monotonicity property.

Fix a query $p \in Q$. Partition the set of all permutations Ω into subsets such that for each subset, all the permutations within the subset have the same order on all queries, except for query p . For any subset Ω_p of this partition, let the permutations be called $\sigma_1, \dots, \sigma_n$, according to the position of p .

Define $alg_\sigma(i, t)$ to be the amount of money that bidder i has spent in the run of $Greedy$ on the permutation σ^3 up to (just after) time t . We have the following version of the **monotonicity lemma**:

LEMMA 3.1. [Monotonicity] *For every $s, s' \in [1, n]$, $s' < s$, for every $t \leq s - 1$, and for every bidder $i \in [1, n]$, we have one of the two conditions:
Either: $alg_{\sigma_{s'}}(i, t + 1) \geq alg_{\sigma_s}(i, t)$*

²As stated in the Introduction, $Greedy$ as such is not monotonic, but we modify $Greedy$ so that it does not truncate the bid by the remaining budget. This variant is the one we analyze here, and is monotonic.

³Note that since the algorithm may overspend budgets, $alg_\sigma(i, t)$ could be greater than B_i (however it is at most $B_i + \epsilon_i$).

Or: $alg_{\sigma_{s'}}(i, t+1) \geq B_i$

Proof. Fix $s' < s \leq n$ and a bidder i . We will prove the lemma by induction on t going from 1 to $s-1$. We are comparing the run of ALG on $\sigma_{s'}$ and on σ_s up to time t . Note, firstly, that the two permutations are identical until position $s'-1$, hence so are the two runs (by the Prefix Property). So, in fact, $\forall t < s', alg_{\sigma_{s'}}(i, t) = alg_{\sigma_s}(i, t)$ and so $alg_{\sigma_{s'}}(i, t+1) \geq alg_{\sigma_s}(i, t)$. Hence we may assume that the base case of the induction is $t = s'-1$. Suppose the statement is true for some $t \in [s'-1, s-2]$. We will show it is true for $t+1$ as well. This would prove the lemma.

Let q be the query $\sigma_s(t+1)$. Since $t \in [s'-1, s-2]$, we know that $\sigma_{s'}(t+2) = q$ as well. Suppose that in the run on σ_s , ALG allocates q to bidder j . We ask: where does ALG allocate q in the run on $\sigma_{s'}$? Define, for a permutation π and a time τ , $avail_{\pi}(\tau)$ as the set of bidders available with non-zero remaining budget in the run of ALG on π at (just before) time τ . By the induction hypothesis, $avail_{\sigma_{s'}}(t+2) \subseteq avail_{\sigma_s}(t+1)$. Also, we know that $j \in avail_{\sigma_s}(t+1)$ and that j is the highest bidder among bidders in $avail_{\sigma_s}(t+1)$.

Now consider two cases: If $j \in avail_{\sigma_{s'}}(t+2)$ then, by the argument above, j is the highest bidder in $avail_{\sigma_{s'}}(t+2)$ as well, and (by the Consistent Tie-breaking Property), ALG will allocate q to j in $\sigma_{s'}$. In the second case, when $j \notin avail_{\sigma_{s'}}(t+2)$, then $alg_{\sigma_{s'}}(j, t+1) \geq B_j$ anyway. Hence, both actions keep the induction statement true for $t+1$.

The prefix property holds as before, by the definition of Greedy:

LEMMA 3.2. [Prefix Property] *For any two permutations σ_1 and σ_2 , and any $t \in [n]$, if we have $\forall s \leq t: \sigma_1^{-1}(s) = \sigma_2^{-1}(s)$, then the runs of Greedy on σ_1 and σ_2 are identical from time 1 to time t .*

For any position s , we will use $tagset_{\sigma}(s, p)$ to mean $tagset_{\sigma}(\sigma(s), p)$. Similarly we define $goodset_{\sigma}(s, p)$ and $badset_{\sigma}(s, p)$. Also define $extra_{\sigma}(s) = extra_{\sigma}(\sigma(s))$. We get the following version of the partition lemma:

LEMMA 3.3. [Partition] *Fix p , and a partition Ω_p .*

$$(1) \quad \forall x \in [1, t]: \\ \cup_{s: x \leq s \leq t+1} goodset_{\sigma_x}(s, p) \supseteq tagset_{\sigma_n}(t, p) \\ (2) \quad \forall x \in [t+1, n]: \quad badset_{\sigma_x}(t, p) = tagset_{\sigma_n}(t, p)$$

Proof. (2): Follows directly from Lemma 3.2.

(1): Suppose i^* be the bidder to which OPT allocated query p . By the Prefix Property, $alg_{\sigma_k}(i^*, k-1) = alg_{\sigma_n}(i^*, k-1)$, and by the monotonicity lemma either $alg_{\sigma_k}(i^*, s+1) \geq B_{i^*}$ or $alg_{\sigma_k}(i^*, s+1) \geq alg_{\sigma_n}(i^*, s)$. So either $alg_{\sigma_k}(i^*, s+1) - alg_{\sigma_k}(i^*, k-1) \geq alg_{\sigma_n}(i^*, s) - alg_{\sigma_n}(i^*, k-1)$, or $alg_{\sigma_k}(i^*, s+1) \geq B_{i^*}$. In either case, $tagset_{\sigma_n}(s, p) \subseteq \bigcup_{k < s' \leq s+1} tagset_{\sigma_k}(s', p)$. But recall that $tagset_{\sigma_k}(s', p)$ contributes towards $goodset_{\sigma_k}(s', p)$ for all s' s.t. $s' \geq k$. This proves the lemma.

Define $good_{\sigma}(s, p) = |goodset_{\sigma}(s, p)|$ and $bad_{\sigma}(s, p) = |badset_{\sigma}(s, p)|$. Also define:

$$miss_{\sigma}(s, p) = \begin{cases} \max(0, opt(p) - alg_{\sigma}(p)) & \text{if } \sigma(s) = p \\ 0 & \text{otherwise.} \end{cases}$$

These three basic properties shown above (Lemmas 3.1, 3.2 and 3.3) suffice to prove the required relationships between $good$, bad and $miss$, and to prove the main theorem. We state these inequalities below (Lemmas 3.5 and 3.6 hold for every $p \in Q$ and every group Ω_p). Their proofs are more involved than the corresponding proofs in the case of matching, and are provided in the Appendix B.

LEMMA 3.4.

$$\forall s \in [n], \forall \sigma \in \Omega: \quad \sum_p bad_{\sigma}(s, p) + \sum_p good_{\sigma}(s, p) \\ + |extra_{\sigma}(s)| + \sum_p miss_{\sigma}(s, p) \geq opt(\sigma(s))$$

LEMMA 3.5.

$$\forall t \in [n] : \quad \text{miss}_{\sigma_t}(t, p) \leq \sum_{\sigma \in \Omega_p} \sum_{s < t} \frac{\text{bad}(s, p)}{n - s}$$

LEMMA 3.6.

$$\forall t \in [n - 1] : \\ \sum_{\sigma \in \Omega_p} \sum_{s \leq t} \text{bad}(s, p) \frac{s}{n - s} \leq \sum_{\sigma \in \Omega_p} \sum_{s \leq t+1} \text{good}(s, p)$$

THEOREM 3.1. *The factor of Greedy in the random-permutation input model is at least $(1 - 1/e)$.*

Sketch of Proof: Lemmas 3.4, 3.5 and 3.6, provide inequalities bounding the variables miss, bad, good and extra. We use the same idea of factor revealing LP (as we used in Section 2) by maximizing the loss of revenue over these inequalities. It turns out that after some rearrangement, we get the same LP (but with our new variables). The details of this proof are provided in Appendix C.

Recall, however, that we had modified Greedy to allow overspending of budgets. We can show (e.g., by induction) that the revenue of this variant of Greedy is almost the same as that of the pure Greedy, which truncates the bids by the remaining budget, when the bids are small (each bidder's spending may be different by at most $n\epsilon$, where epsilon is the largest bid). Thus we have proved the factor for both versions of Greedy.

4 Additional Results

4.1 Tightness of the analysis We prove that our analysis is tight, by giving an example in which Greedy does no better than $1 - 1/e$ (details are given in Appendix D):

THEOREM 4.1. *The factor of Greedy in the random-permutation input model is no more than $(1 - 1/e)$.*

4.2 The *i.i.d.* randomized input model In this model there is a fixed (but unknown) distribution on Q , and the next query in the sequence is picked independently from this distribution. A simple reduction to our random permutation setting shows that:

THEOREM 4.2. *The factor of Greedy in the independent distribution input model is at least $(1 - 1/e)$. Also, there is an example distribution over which Greedy does no better than $1 - 1/e$.*

Proof. (Sketch) In this model we have a sample space consisting of the different query sequences obtained by drawing n times from the given distribution. We have to compare the expected performance of Greedy with the expected performance of OPT over this sample space⁴. Divide the sample space into classes s.t. the set of queries is the same for every sequence in a class. Clearly, each class consists of all the permutations of some set, and furthermore, the probability of occurrence of each sequence in a class is the same. Suppose Ω' is one such class. Then, by Theorem 3.1 we get that $E[\text{GREEDY}|\Omega'] \geq (1 - 1/e)E[\text{OPT}|\Omega']$. Taking expectation over the different classes, we get the first part of the theorem. For the second part, a tight example is provided in Appendix D.1.

4.3 Lower Bound The proof of the following theorem uses a uniform distribution on a set of simple 3×3 size bipartite matching examples, and can be found in Appendix E.

THEOREM 4.3. *No deterministic algorithm can have a competitive ratio better than $3/4$, and no randomized algorithm can have a competitive ratio better than $5/6$, for the case of bipartite matching in the random permutations model.*

⁴The result also holds for the average of the ratios of Greedy and OPT, and not only for the ratio of their averages.

5 Further Discussion

Our motivation for studying Greedy was that it is a natural algorithm in an auction style process. However, the question naturally arises: Is Greedy optimal in our distributional model? If we make the assumption that there are n types of keywords, and many queries of each type, and that we know the length of the query sequence, then there is an allocation algorithm with a ratio very close to 1: Sample the sequence for an ϵ fraction of its length and learn the distribution of queries. Now solve and round a revenue maximization Linear Program to get $1 - \epsilon$ ratio allocation. However, this sampling method will not work if all the queries are distinct, and of course, this method could be arbitrarily bad in the worst case. We do not know of any algorithm provably better than Greedy in the random permutation model, but we believe that the algorithm from [MSVV05] performs strictly better. Currently we can show that it cannot do better than a factor $\alpha \sim 0.81$.

Recall that in the case of matching, it was proved in [KVV90] that RANKING in the worst case and Greedy in the random permutations case are duals of each other. The question naturally arises for the general assignment problem: What is the dual algorithm to Greedy? It turns out that no such duality result seems to hold in the general case. Furthermore any such dual would have the bidders arriving in online, and hence would not apply to online assignment.

Finally, it is important to investigate the extent to which the RANKING algorithm of [KVV90] and its ideas can be generalized, maybe even to offline algorithms. Our problem can be considered to be a special case of the problem of combinatorial auctions with submodular utility bidders. In this setting an important problem is to find offline algorithms using only value queries [LLN01]. There exist algorithms with factor $1 - 1/e$ [DS06, Fei06] (and even slightly better [FV06]) but using the stronger demand queries, and there is a gap between the factor and the hardness in the value query model. Can the ideas in our algorithm help close this gap? Even the case of Adwords with large bids is open (recall that the best known offline algorithms gives factor $1 - 1/e$ [AM04]). The same question can be asked for the 0-1-XOS utility model (also known as the MCG problem [CK04]). We believe that our techniques may be useful for some of these problems.

References

- [AGM06] Gagan Aggarwal, Ashish Goel, and Rajeev Motwani. Truthful auctions for pricing search keywords. In *ACM Conference on Electronic Commerce*, pages 1–7, 2006.
- [AM04] N. Andelman and Y. Mansour. Auctions with budget constraints. In *9th Scandinavian Workshop on Algorithm Theory (SWAT)*, pages 26–38, 2004.
- [BIK07] M. Babaioff, N. Immorlica, and R. Kleinberg. Matroids, secretary problems, and online mechanisms. In *SODA*, 2007.
- [CK04] C. Chekuri and A. Kumar. Maximum coverage problem with group budget constraints and applications. In *APPROX*, 2004.
- [DS06] Shahar Dobzinski and Michael Schapira. An improved approximation algorithm for combinatorial auctions with submodular bidders. In *SODA*, 2006.
- [Dyn63] E. B. Dynkin. The optimum choice of the instant for stopping a markov process. *Sov. Math. Dokl.*, 4, 1963.
- [EOS07] B. Edelman, M. Ostrovsky, and M. Schwarz. Internet advertising and the generalized second price auction: Selling billions of dollars worth of keywords. *American Economic Review*, 97(1):242–259, 2007.
- [Fei06] Uriel Feige. On maximizing welfare when utility functions are subadditive. In *STOC*, 2006.
- [FGMS06] Lisa Fleischer, Michel X. Goemans, Vahab S. Mirrokni, and Maxim Sviridenko. Tight approximation algorithms for maximum general assignment problems. In *SODA*, pages 611–620, 2006.
- [FMPS07] J. Feldman, S. Muthukrishnan, M. Pal, and C. Stein. Budget optimization in search-based advertising auctions. In *ACM Conference on Electronic Commerce*, 2007.
- [FV06] Uriel Feige and Jan Vondrak. Approximation algorithms for allocation problems: Improving the factor of $1-1/e$. In *FOCS*, 2006.
- [GM06] Sudipto Guha and Andrew McGregor. Approximate quantiles and the order of the stream. In *PODS*, pages 273–279, 2006.
- [KV07] Erik Krohn and Kasturi Varadarajan. *Private Communication*, 2007.
- [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.

- [LPSV07] S. Lahaie, D. Pennock, A. Saberi, and R. Vohra. *Algorithmic Game Theory (Nisan et al. editors)*, chapter Sponsored Search. 2007.
- [MNS07] M. Mahdian, H. Nazerzadeh, and A. Saberi. Allocating online advertisement space with unreliable estimates. In *ACM Conference on Electronic Commerce*, 2007.
- [MSVV05] Aranyak Mehta, Amin Saberi, Umesh Vazirani, and Vijay Vazirani. Adwords and generalized online matching. In *FOCS*, 2005.
- [RW06] P. Rusmevichientong and D. P. Williamson. An adaptive algorithm for selecting profitable keywords for search-based advertising services. In *Seventh ACM Conference on Electronic Commerce*, 2006.
- [Var06] H. Varian. Position auctions. *International Journal of Industrial Organization (To appear)*, 2006.
- [Yao77] A. C. Yao. Probabilistic computations: towards a unified measure of complexity. *FOCS*, pages 222–227, 1977.

A Aggregating the inequalities: A Factor Revealing LP

Now we will aggregate the constraints in Lemma 2.2 and 2.3 over all the permutations and girls. For that, we need to first define the following:

$$\begin{aligned} miss(s) &= E_{\sigma \in \Omega} \left[\sum_p miss_{\sigma}(s, p) \right], \\ bad(s) &= E_{\sigma \in \Omega} \left[\sum_p bad_{\sigma}(s, p) \right], \\ good(s) &= E_{\sigma \in \Omega} \left[\sum_p good_{\sigma}(s, p) \right] \end{aligned}$$

By summing the constraints in Lemma 2.2 over all permutations classes Ω_p and all p , we get

$$\forall t : \sum_p \sum_{\sigma \in \Omega} miss_{\sigma}(t, p) \leq \sum_p \sum_{\sigma \in \Omega} \sum_{s:s < t} bad_{\sigma}(s, p) / (n - s)$$

Changing the order of summation and dividing by $|\Omega|$, we get,

COROLLARY A.1.

$$\forall t : \quad miss(t) \leq \sum_{s:s < t} bad(s) / (n - s)$$

Similarly we can get,

COROLLARY A.2.

$$\forall t : \quad \sum_{s \leq t} bad(s) \frac{s}{n - s} \leq \sum_{s:s \leq t+1} good(s)$$

Proof of Theorem 2.1 : From Lemma 2.1 and Corollaries A.1 and A.2 we have:

$$\begin{aligned} \text{Min: } & \sum_t [bad(t) + good(t)] \\ \text{s.t.: } & \forall t < n : \quad \sum_{s \leq t+1} good(s) - \sum_{s \leq t} bad(s) \frac{s}{n-s} \geq 0 \\ & \forall t \leq n : \quad good(t) + bad(t) + \sum_{s < t} \frac{bad(s)}{n-s} \geq 1 \\ & \quad \quad \quad good(t), bad(t), miss(t) \geq 0 \end{aligned} \tag{1.3}$$

Now change variables as follows: $m_t = good(t) + bad(t)$, $v_t = bad(t)/(n - t)$

$$\text{Min: } \sum_t m_t$$

$$\begin{aligned}
\text{s.t.: } \sum_{s \leq t+1} m_s - n \sum_{s \leq t} v_s &\geq 0 \\
m_t + \sum_{s < t} v_s &\geq 1 \\
m_t, v_t &\geq 0
\end{aligned}$$

It can be shown that dropping the term m_{t+1} from the first set of equations does not affect the LP solution by much. A proof of a similar statement is in KVV, and so are the calculations. We provide an LP based calculation here, for completeness. We get the following primal-dual pair:

$$\begin{array}{ll}
\text{Min: } \sum_t m_t & \text{Max: } \sum_t \alpha_t \\
\text{s.t.:} & \text{s.t.:} \\
\sum_{s \leq t} m_s - n \sum_{s \leq t} v_s \geq 0 & \sum_{s > t} \alpha_s - n \sum_{s \geq t} \beta_s \leq 0 \\
m_t + \sum_{s < t} v_s \geq 1 & \alpha_t + \sum_{s \geq t} \beta_s \leq 1 \\
m_t, v_t \geq 0 & \alpha_t, \beta_t \geq 0
\end{array}$$

Both LPs have the property that if we set all inequalities tight, we get a feasible solution. This means that these solutions are optimal.

$$m_t = \alpha_{n+1-t} = (1 - 1/n)^{i-1}, v_t = \beta_{n+1-t} = \frac{1}{n}(1 - 1/n)^{i-1}$$

The optimal objective function sums up to $n(1 - 1/e)$ proving the theorem⁵. □

B Proofs of the Main Lemmas from Section 3

Proof of Lemma 3.4 : For every σ , and every $s \in [1, n]$,

$$\begin{aligned}
(2.4) \quad \sum_p [bad_\sigma(s, p) + good_\sigma(s, p)] &= \sum_p |tagset_\sigma(\sigma(s), p)| \\
&= alg_\sigma(\sigma(s)) - |extra_\sigma(s)|
\end{aligned}$$

The first equality is because of the fact that p is either above or below s in σ . The second equality follows from the definition of $extra_\sigma(s)$. Now, by definition, $miss_\sigma(s, p)$ is possibly non-zero only for $p = \sigma(s)$, in which case it is equal to $\max\{0, opt(p) - alg_\sigma(p)\}$. Hence,

$$\sum_p miss_\sigma(s, p) = \max\{0, opt(\sigma(s)) - alg_\sigma(\sigma(s))\}$$

From (2.4), we get that

$$\sum_p miss_\sigma(s, p) = \max \left\{ 0, opt(\sigma(s)) - \sum_p bad_\sigma(s, p) - \sum_p good_\sigma(s, p) - |extra_\sigma(s)| \right\}$$

thus proving the lemma. □

Proof of Lemma 3.5 : Define the quantity $x := \sum_{s < t} bad_{\sigma_n}(s, p)$. By the definition of $bad_{\sigma_n}(s, p)$, $x \leq opt(p)$. In fact, we shall show the following inequality:

⁵We assumed that the optimal matching is perfect, i.e. $OPT = n$. The case of $OPT < n$ can be taken care of with minor changes in the proof.

$$(2.5) \quad \text{miss}_{\sigma_t}(t, p) \leq x = \sum_{\sigma \in \Omega_p} \sum_{s < t} \frac{\text{bad}_{\sigma}(s, p)}{n - s}$$

We prove Equation (2.5) in a sequence of two claims:

CLAIM 1. $\text{miss}_{\sigma_t}(t, p) \leq x$

Proof: Let i_p^* be the bidder to whom OPT allocated the query p . We show the following subclaim:

Subclaim: Consider the run on the permutation σ_t . At time t (when ALG considers assigning query p), the remaining budget of i_p^* is at least $\text{opt}(p) - x \geq 0$.

Proof: First note that, by the Prefix Property, $\sum_{s < t} \text{bad}_{\sigma_t}(s, p) = \sum_{s < t} \text{bad}_{\sigma_n}(s, p) =: x$. Since in σ_t , $\forall s < t$, $\text{bad}_{\sigma_t}(s, p) = |\text{tagset}_{\sigma_t}(\sigma_t(s), p)|$, therefore $\sum_{s < t} |\text{tagset}_{\sigma_t}(\sigma_t(s), p)| = x$. But this means that at least $\text{opt}(p) - x$ length of $\text{Opt-Interval}(p)$ is untagged after time $t - 1$ in the run on σ_t . Of course, this means that at least $\text{opt}(p) - x$ amount of budget of i_p^* is still unspent at this time. This proves the subclaim.

If $x < \text{opt}(p)$, then using the fact that the algorithm is greedy (and that we do not truncate the bids of bidders by their remaining non-zero budgets), we see that $\text{alg}(p) \geq \text{opt}(p)$. If $x = \text{opt}(p)$, then $\text{alg}(p) \geq \text{opt}(p) - x$, trivially. This proves the claim.

CLAIM 2. $\sum_{\sigma \in \Omega_p} \sum_{s < t} \frac{\text{bad}_{\sigma}(s, p)}{n - s} = x$

Proof: Fix an $s < t$, and consider the value of $\sum_{\sigma \in \Omega_p} \frac{\text{bad}_{\sigma}(s, p)}{n - s}$. Note that, by the definition of $\text{bad}_{\sigma}(s, p)$ and by the Prefix Property:

$$\text{bad}_{\sigma_k}(s, p) = \begin{cases} \text{bad}_{\sigma_n}(s, p) & \text{for } k \in [s + 1, n] \\ 0 & \text{for } k \in [1, s] \end{cases}$$

Hence $\frac{\sum_{\sigma \in \Omega_p} \text{bad}_{\sigma}(s, p)}{n - s} = \text{bad}_{\sigma_n}(s, p)$. Hence, $\sum_{\sigma \in \Omega_p} \sum_{s < t} \frac{\text{bad}_{\sigma}(s, p)}{n - s} = \sum_{s < t} \text{bad}_{\sigma_n}(s, p) = x$.

This concludes the proof of Equation (2.5), and hence of the lemma.

□

Proof of Lemma 3.6 : To do a tight counting for the proof of the lemma, we need to define a notion of weighted set. Given a universe U of elements, a weighted set S_w has weight on all its elements. In particular, an unweighted set S can be viewed as a weighted set with weight one on elements which are present in the set, and zero otherwise. We shall also define two new operations $*$ and \oplus on these weighted sets. Given a weighted set S , and a number x , we define new weighted set $S' = S * x$ to mean that $\text{weight}_{S'}(e) = \text{weight}_S(e) * x, \forall e \in U$. Also given weighted sets S_1 and S_2 , we define $S' = S_1 \oplus S_2$ to mean that $\text{weight}_{S'}(e) = \text{weight}_{S_1}(e) + \text{weight}_{S_2}(e)$. $S_1 \subseteq_w S_2$ means that $\text{weight}_{S_1}(e) \leq \text{weight}_{S_2}(e), \forall e \in U$. Define the measure of weighted set S , $\mu(S) = \int_U \text{weight}_S(u) du$. For the rest of the proof, the term set will mean a weighted set. We will consider sets on the universe $\text{Opt-Interval}(p)$, which, recall, is a interval of size $\text{opt}(p)$, corresponding to a query p .

Fix an $s \leq t$. Consider a subset Ω_p s.t. $\text{badset}_{\sigma_n}(s, p) \neq \emptyset$. By the Prefix Property, $\text{badset}_{\sigma_i}(s, p) = \text{badset}_{\sigma_n}(s, p), \forall i \in [s + 1, n]$. By Partition lemma, Lemma 3.3, we see that for every $k \leq s, \forall l \in [s + 1, n]$: $\text{badset}_{\sigma_l}(s, p) \subseteq \bigcup_{s'=k}^{s+1} \text{goodset}_{\sigma_k}(s', p)$, where the union is a disjoint union.

Since the union is disjoint, this is the same as

$$\forall k \in [1, s], \forall l \in [s + 1, n]: \quad \text{badset}_{\sigma_l}(s, p) \subseteq_w \bigoplus_{s'=k}^{s+1} \text{goodset}_{\sigma_k}(s', p)$$

By averaging, we see that:

$$\forall s: \quad \frac{1}{n - s} \bigoplus_{l=s+1}^n \text{badset}_{\sigma_l}(s, p) \subseteq_w \frac{1}{s} \bigoplus_{k=1}^s \bigoplus_{s'=k}^{s+1} \text{goodset}_{\sigma_k}(s', p)$$

or,

$$\forall s : \bigoplus_{l=s+1}^n \text{badset}_{\sigma_l}(s, p) \frac{s}{n-s} \subseteq_w \bigoplus_{k=1}^s \bigoplus_{s'=k}^{s+1} \text{goodset}_{\sigma_k}(s', p)$$

But since $\text{badset}_{\sigma_i}(s, p) = \phi$, $\forall i \in [1, s]$ and $\forall s' < k$: $\text{goodset}_{\sigma_k}(s', p) = \phi$, we get:

$$\forall s : \bigoplus_{l=1}^n \text{badset}_{\sigma_l}(s, p) \frac{s}{n-s} \subseteq_w \bigoplus_{k=1}^s \bigoplus_{s'=1}^{s+1} \text{goodset}_{\sigma_k}(s', p)$$

And again, since $\forall k > s, s' \leq s$: $\text{goodset}_{\sigma_i}(s', p) = \phi$, we get :

$$\forall s : \bigoplus_{l=1}^n \text{badset}_{\sigma_l}(s, p) \frac{s}{n-s} \subseteq_w \bigoplus_{k=1}^n \bigoplus_{s'=1}^{s+1} \text{goodset}_{\sigma_k}(s', p)$$

or,

$$\forall s : \bigoplus_{\sigma \in \Omega_p} \text{badset}_{\sigma}(s, p) \frac{s}{n-s} \subseteq_w \bigoplus_{\sigma \in \Omega_p} \bigoplus_{s'=1}^{s+1} \text{goodset}_{\sigma}(s', p)$$

Also, by definition, $\forall u \neq s, u, s \leq t$: $\text{badset}_{\sigma_n}(s, p) \cap \text{badset}_{\sigma_n}(u, p) = \emptyset$. Hence:

$$\bigoplus_{s \leq t} \bigoplus_{\sigma \in \Omega_p} \text{badset}_{\sigma}(s, p) \frac{s}{n-s} \subseteq_w \bigoplus_{\sigma \in \Omega_p} \bigoplus_{s \leq t+1} \text{goodset}_{\sigma}(s, p)$$

Therefore,

$$\begin{aligned} \mu \left(\bigoplus_{s \leq t} \bigoplus_{\sigma \in \Omega_p} \text{badset}_{\sigma}(s, p) \frac{s}{n-s} \right) &\leq \mu \left(\bigoplus_{s \leq t+1} \bigoplus_{\sigma \in \Omega_p} \text{goodset}_{\sigma}(s, p) \right) \\ \sum_{s \leq t} \sum_{\sigma \in \Omega_p} \mu(\text{badset}_{\sigma}(s, p)) \frac{s}{n-s} &\leq \sum_{s \leq t+1} \sum_{\sigma \in \Omega_p} \mu(\text{goodset}_{\sigma}(s, p)) \\ \sum_{s \leq t} \sum_{\sigma \in \Omega_p} \text{bad}_{\sigma}(s, p) \frac{s}{n-s} &\leq \sum_{s \leq t+1} \sum_{\sigma \in \Omega_p} \text{good}_{\sigma}(s, p) \end{aligned}$$

□

C Proof of the Main Theorem - Theorem 3.1

We shall use the same definition of $\text{miss}(s)$, $\text{bad}(s)$ and $\text{good}(s)$ as defined in Appendix A. Let $\text{extra}(s) = E_{\sigma \in \Omega} |\text{extra}_{\sigma}(s)|$. Using the similar approach in Appendix A, we will get the following three corollaries from Lemmas 3.4, 3.5 and 3.6.

COROLLARY C.1.

$$\forall t : \quad \text{bad}(t) + \text{good}(t) + \text{extra}(t) + \text{miss}(t) \geq \text{OPT}/n$$

COROLLARY C.2.

$$\forall t : \quad \text{miss}(t) \leq \sum_{s:s < t} \text{bad}(s)/(n-s)$$

COROLLARY C.3.

$$\forall t : \quad \sum_{s \leq t} \text{bad}(s) \frac{s}{n-s} \leq \sum_{s:s \leq t+1} \text{good}(s)$$

Using the above corollaries, we get the following LP:

$$\begin{aligned}
\text{Min: } & \sum_t [\text{bad}(t) + \text{good}(t) + \text{extra}(t)] \\
\text{s.t.: } & \forall t < n: \quad \sum_{s \leq t+1} \text{good}(s) - \sum_{s \leq t} \text{bad}(s) \frac{s}{n-s} \geq 0 \\
& \forall t \leq n: \quad \text{good}(t) + \text{bad}(t) + \text{extra}(t) + \sum_{s < t} \frac{\text{bad}(s)}{n-s} \geq \text{OPT}/n \\
& \text{good}(t), \text{bad}(t), \text{extra}(t) \geq 0
\end{aligned}$$

First inequality can in fact be relaxed to $\forall t < n: \sum_{s \leq t+1} (\text{good}(s) + \text{extra}(s)) - \sum_{s \leq t} \text{bad}(s) \frac{s}{n-s} \geq 0$.

Now substituting m_t for $\text{good}(t) + \text{bad}(t) + \text{extra}(t)$, v_t for $\text{bad}(t)/n - t$, and solving it similar to Appendix A, we get that the value of objective function is at least $\text{OPT}(1 - 1/e)$.

D A Tight Example - Proof of Theorem 4.1

In this section we provide an example to show that the Greedy algorithm can do no better than $1 - 1/e$ in the random-permutation input model, and thus exhibit that our analysis is tight. The example is similar to the one used in [MSVV05] to show a lower-bound on randomized algorithms in the online worst-case model.

In this instance, there are N bidders, each with a budget of L (a large integer). The queries are grouped into N groups, Q_1, Q_2, \dots, Q_N . Each group has L identical queries. Each query in Q_i gets the following bids: Bidders 1 through $i - 1$ bid 0; bidders i through N bid 1. Clearly, $\text{OPT} = LN$, by allocating, for all i , all the queries in Q_i to bidder i .

We will assume that Greedy breaks ties in the following way: when an query q arrives, it will be allocated to the highest numbered bidder who bids 1 and has available budget. The assumption about bad tie-breaking can be removed by a simple perturbation of the bids: for example, for all i , replace all the 1-bids of bidder i by $1 + \epsilon_i$, where $0 < \epsilon_1 < \epsilon_2 < \dots < \epsilon_N \ll 1$. The budgets can be adjusted accordingly.

Now consider the performance of Greedy on the sequence of these queries arriving in a random permutation σ . Clearly, the first L queries in σ will all be allocated to bidder N (who bids for all queries). Among the next L queries in σ , all will be allocated to bidder $N - 1$, except for any queries from Q_N , which are left unallocated. Now among the third L queries in σ , some are allocated to $N - 1$, most to $N - 2$ and some are lost. Let us instead count in a different manner: It is clear that bidder N , by symmetry, is allocated at most L/N queries from Q_j , in expectation, $\forall j$. Similarly, bidder $N - 1$, by symmetry, is allocated at most $L/(N - 1)$ queries from Q_j , in expectation, $\forall j \leq N - 1$. In this manner one can see that bidder i is allocated at most L/i queries from Q_j , in expectation, $\forall j \leq i$.

Counting differently, for $i = 1, \dots, N$, the number of queries in Q_i that get allocated is $\min\{L, \sum_{j \leq i} \frac{L}{N-j}\}$. Summing this up, we can show that the expected amount of money spent by Greedy is $LN(1 - 1/e) = (1 - 1/e)\text{OPT}$.

We note that this analysis is very similar to that in [MSVV05] for the lower bound in the worst case online setting, the two being a form of duals of each other. The difference is that the role of bidders and query groups is reversed – here it is the last $N(1 - 1/e)$ bidders who finish most of their budgets, while in [MSVV05] it was the first $N(1 - 1/e)$ query groups that get almost completely allocated.

D.1 Tight example in the i.i.d. model Recall that in the i.i.d. model we have an unknown distribution over keywords, and the next query in the sequence is picked independently from this distribution. In our example there are N bidders. Consider a uniform distribution over N keywords, where the i^{th} keyword gets the following bids: Bidders 1 through $i - 1$ bid 0; bidders i through N bid 1. We will use the tie breaking rules as discussed in Appendix D. We will sample LN queries from the above distribution. Now by the Chernoff bound, we can show that, for all δ , there is a large enough L so that w.h.p. each keyword is sampled at most $L + \delta/N$ times. Now take the sample space in which each keyword occurs at least $L - \delta/N$ times, and divide it into families s.t. in each family, every sequence has a fixed set of queries. Using arguments similar to the ones in Appendix D, we can show that expected performance of greedy, conditional over any such family, is at most $LN(1 - 1/e) + \delta$. Now taking expectation over all such families we get that expected performance of greedy is at most $LN(1 - 1/e) + \delta$.

E The Lower Bound - Proof of Theorem 4.3

(Sketch of Proof) For the bound on deterministic algorithms, consider a very simple 2 boy, 2 girl setting, where boy 1 could be matched to both the girls, but boy 2 can only be matched to one of the two girls. By looking at the “code” of the deterministic algorithm, we can decide which girl boy 2 can be matched to, so that in one of the two random permutations of the boys, only one pair can be matched. This shows a factor of $3/4$.

For the lower bound on randomized algorithms, we will use Yao’s Lemma [Yao77] to prove this statement. We start with a distribution on inputs of the following form: Take an $n \times n$ complete upper triangular matrix with $0, 1$ entries. This is the incident matrix of the bipartite graph, with the rows being the girls and columns being boys. We use the uniform distribution over different inputs corresponding to all permutations of the row names. As according to the model, each input has the columns arrive in a random permutation. Thus the distribution is essentially uniform distribution on both row and column permutations of the complete upper triangular matrix.

As opposed to the proof in [KVV90] for the worst case input model, it turns out to be difficult to characterize the performance of *any* deterministic algorithm on this input distribution. This is because a deterministic algorithm can (in this model) be smart and use a lot of information from the past (as a very simple example, if it sees a column with $n - 1$ 1’s and then a column with n 1’s then it knows that the extra row is the row with n 1’s). We have not been able to come up with a general analysis for n , but we can do a brute force analysis for $n = 3$, in which we can take care of how any deterministic algorithm may use all such information from the past. The brute force analysis shows that the performance of any deterministic algorithm over this input distribution is no more than $5/6$, thus proving a lower bound of $5/6$ for randomized algorithms in the random permutations model.