Towards a Practical, Budget-Oblivious Algorithm for the Adwords Problem Under Small Bids

Vijay V. Vazirani ✉
University of California, Irvine, CA, USA

Abstract

Motivated by recent insights into the online bipartite matching problem (OBM), our goal was to extend the optimal algorithm for it, namely Ranking, all the way to the special case of adwords problem, called Small, in which bids are small compared to budgets; the latter has been of considerable practical significance in ad auctions [20]. The attractive feature of our approach was that it would yield a *budget-oblivious algorithm*, i.e., the algorithm would not need to know budgets of advertisers and therefore could be used in autobidding platforms.

We were successful in obtaining an optimal, budget-oblivious algorithm for Single-Valued, under which each advertiser can make bids of one value only. However, our next extension, to Small, failed because of a fundamental reason, namely failure of the *No-Surpassing Property*. Since the probabilistic ideas underlying our algorithm are quite substantial, we have stated them formally, after assuming the No-Surpassing Property, and we leave the open problem of removing this assumption.

With the help of two undergrads, we conducted extensive experiments on our algorithm on randomly generated instances. Our findings are that the No-Surpassing Property fails less than 2% of the time and that the performance of our algorithms for Single-Valued and Small are comparable to that of [20]. If further experiments confirm this, our algorithm may be useful as such in practice, especially because of its budget-obliviousness.

2012 ACM Subject Classification Theory of computation → Algorithmic mechanism design

Keywords and phrases Adwords problem, ad auctions, online bipartite matching, competitive analysis

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2023.21

Related Version *Full Version*: `https://arxiv.org/pdf/2107.10777.pdf`

Funding *Vijay V. Vazirani*: Supported in part by NSF grant CCF-2230414.

1 Introduction

The *adwords problem*, called Adwords[1] in this paper, involves matching keyword queries, as they arrive online, to advertisers; the latter have daily budget limits and they make bids for the queries. Its special case when bids are small compared to budgets, called Small in this paper, captures a key computational issue that arises in the context of ad auctions, for instance in Google's AdWords marketplace. An optimal algorithm for Small, achieving a competitive ratio of $\left(1-\frac{1}{e}\right)$, was first given in [20]; for the impact of this result in the marketplace, see Section 1.2. *In this paper, we give a new budget-oblivious online algorithm for Small.*

A *budget-oblivious online algorithm* does not know the daily budgets of advertisers; however, in a run, it knows when the budget of an advertiser is exhausted. Yet its revenue is compared to the optimal revenue generated by an offline algorithm with full knowledge of the budget. The importance of a budget-oblivious algorithm lies in its use in autobidding

[1] For formal statements of problems studied in this paper, see Section 2

platforms [1, 6], which manage the ad campaigns of large advertisers; they dynamically adjust bids and budgets over multiple search engines to improve performance. In Open Problem Number 20, Mehta [19] asks for such an algorithm for SMALL.

Recent insights on the online bipartite matching problem (OBM) encouraged us to seek such an algorithm. A simple optimal algorithm, called RANKING, achieving a competitive ratio of $\left(1 - \frac{1}{e}\right)$, was given in [17] for OBM. However, the analysis of RANKING given in [17] was difficult to comprehend. A sequence of papers has finally led to a simple and elegant analysis, see Section 1.1. The simplicity of RANKING is particularly attractive; moreover, it has become the paradigm-setting algorithmic idea in the area of online and matching-based market design [8].

Ideas underlying the new proof of OBM enabled us to generalize RANKING all the way to an algorithm for SMALL, while retaining the simplicity of the RANKING. As a result of this simplicity, our algorithm has better properties than [20]; in particular, it is budget-oblivious. A detailed discussion of its running time is given below. A budget-oblivious algorithm for SMALL, having a competitive ratio of 0.522[2] was recently obtained by Udwani [21], using the idea of an LP-free analysis, which involves writing appropriate linear inequalities to compare the online algorithm with the offline optimal algorithm.

At the outset of this work, extending RANKING directly to SMALL seemed an uphill task. Therefore we attempted an intermediate problem first, namely SINGLE-VALUED, in which each advertiser can make bids of one value only, although the value may be different for different advertisers. We note that [2] had already obtained an optimal online algorithm for SINGLE-VALUED by reducing it to the vertex weighted online matching problem, see Section 1.1 for details. As explained in Section 1.3, in order to develop tools for attacking SMALL, we needed to solve SINGLE-VALUED *directly*, and not resort to this reduction.

Our algorithm for SINGLE-VALUED is optimal, and it is also budget-oblivious. Furthermore, our algorithm uses fewer random bits than the approach of [2]; see Section 1.1 for a detailed comparison. We note that in contemporary[3] and independent work, Albers and Schubert [3] obtained an identical result for SINGLE-VALUED; their technique is different and involves formulating a configuration LP and conducting a primal-dual analysis. Our technical ideas are described in Section 1.3.

Our analysis of SINGLE-VALUED involved new ideas from two domains, namely probability theory and combinatorics, with the former playing a dominant role and the latter yielding a proof of a condition called the *No-Surpassing Property*, see Property 11. Equipped with these ideas, we next attempted an extension from RANKING to SMALL. Although we met with success in extending the more difficult, probabilistic part, of the argument, we found a counter-example to the combinatorial part, showing that the No-Surpassing Property does not hold for SMALL.

In order to make the no-surpassing property fail, we had to intricately "doctored up" the instance of SMALL. This raised the question of experimentally determining how often this property fails in typical instances and how it affects the performance of our algorithm; for the latter, we compared it to [20]. As can be seen in the four Tables in the full paper [24], the property fails rarely, for less than 2% of the edges (i, j), and the performance of our algorithms for SINGLE-VALUED and SMALL are comparable with that of the MSVV Algorithm. For this reason, and because of its budget-obliviousness, the algorithm may be useful as such in practice. Clearly, it will be good to obtain further experimental confirm on varied types of instances.

[2] Note that the greedy algorithm, which is clearly budget-oblivious, achieves a competitive ratio of 0.5.
[3] Our paper was first posted on arXiv on July 22, 2021 [22].

Since the ideas underlying our algorithm for SMALL, and the probabilistic part of its proof, are quite substantial, we have stated them formally, after assuming the No-Surpassing Property, see the full paper [24]. Under this assumption, we prove a competitive ratio of $\left(1-\frac{1}{e}\right)$ for our algorithm. The problem of obtaining a tight unconditional competitive ratio of our algorithm is an important one and has received much attention over the last two years, ever since the appearance of this paper on arXiv. Critical insights into this open problem are provided by the following results: first, Udwani [21] gave an example to show that the unconditional competitive ratio of our algorithm is strictly less than $(1-1/e)$. Next, Liang et al. [18] showed that the unconditional competitive ratio is less than 0.624; in contrast, $(1-1/e) \approx 0.632$.

▶ Remark 1. The objective of all problems studied in this paper is to maximize the total revenue accrued by the online algorithm. In economics, such a solution is referred to as *efficient*, since the amount bid by an advertiser is indicative of how useful the query is to it, and hence to the economy.

1.1 Related Works

OBM occupies a central place not only in online algorithms but also in matching-based market design, see details in Section 1.2. The analysis of RANKING given in [17] was considered "difficult" and it also had an error. Over the years, several researchers contributed valuable ideas to simplifying its proof. The first simplifications, in [11, 4], got the ball rolling, setting the stage for the substantial simplification given in [7], using a randomized primal-dual approach. [7] introduced the idea of splitting the contribution of each matched edge into primal and dual contributions and lower-bounding each part separately. Their method for defining prices p_j of goods, using randomization, was used by subsequent papers, including this one[4].

Interestingly enough, the next simplification involved removing the scaffolding of LP-duality and casting the proof in purely probabilistic terms[5], using notions from economics to split the contribution of each matched edge into the contributions of the buyer and the seller. This elegant analysis was given by [9]. We note that when we move to generalizations of OBM, even this economic interpretation needs to be dropped, see Remark 4. Building on these works, and incorporating a further simplification relating to the No-Surpassing Property for OBM, a "textbook quality" proof was recently given in [23].

An important generalization of OBM is online b-matching. This problem is a special case of ADWORDS in which the budget of each advertiser is $\$b$ and the bids are 0/1. [16] gave a simple optimal online algorithm, called BALANCE, for this problem. BALANCE awards the next query to the interested bidder who has been matched least number of times so far. [16] showed that as b tends to infinity, the competitive ratio of BALANCE tends to $\left(1-\frac{1}{e}\right)$.

Observe that b-matching is a special case of SMALL, if b is large. Indeed, MSVV Algorithm was obtained by extending BALANCE[6] as follows: [20] first gave a simpler proof of the competitive ratio of BALANCE using the notion of a *factor-revealing LP* [15]. Then they

[4] For a succinct proof of optimality of the underlying function, e^{x-1}, see Section 2.1.1 in [12].
[5] Even though there is no overt use of LP-duality in the proof of [9], it is unclear if this proof could have been obtained directly, without going the LP-duality-route.
[6] It is worth recalling that [20] had first attempted extending OBM to SMALL; however, in the absence of new insights into OBM, this did not go very far.

gave the notion of a *tradeoff-revealing LP*, which yielded an algorithm achieving a competitive ratio of $\left(1-\frac{1}{e}\right)$. [20] also proved that this is optimal for b-matching, and hence SMALL, by proving that no randomized algorithm can achieve a better ratio for online b-matching; previously, [16] had shown a similar result for deterministic algorithms.

The MSVV Algorithm is simple and operates as follows. The effective bid of each bidder j for a query is its bid multiplied by $(1-e^{L_j/B_j})$, where B_j and L_j are the total budget and the leftover budget of bidder j, respectively; the query is matched to the bidder whose effective bid is highest. As a result, the MSVV Algorithm needs to know the total budget of each bidder. Following [20], a second optimal online algorithm for SMALL was given in [5], using a primal-dual approach.

Another relevant generalization of OBM is online vertex weighted matching, in which the offline vertices have weights and the objective is to maximize the weight of the matched vertices. [2] extended RANKING to obtain an optimal online algorithm for this problem. Clearly, SINGLE-VALUED is intermediate between ADWORDS and online vertex weighted matching. [2] gave an optimal online algorithm for SINGLE-VALUED by reducing it to online vertex weighted matching. This involved creating k_j copies of each advertiser j. As a result, their algorithm needs to use $\sum_{j\in A} k_j$ random numbers, where A is the set of advertisers. On the other hand, our algorithm, and that of [3], needs to use only $|A|$ numbers.

ADWORDS is a notoriously difficult problem, partly due to its inherent structural difficulties, which are described in the full paper [24]. For ADWORDS, the greedy algorithm, which matches each query to the highest bidder, achieves a competitive ratio of 1/2. Until recently, that was the best possible. In [13] a marginally improved algorithm, with a ratio of 0.5016, was given. It is important to point out that this 60-page paper was a tour-de-force, drawing on a diverse collection of ideas – a testament to the difficulty of this problem.

In the decade following the conference version (FOCS 2005) of [20], search engine companies generously invested in research on models derived from OBM and adwords. The reason was two-fold: the substantial impact of [20] and the emergence of a rich collection of digital ad tools. It will be impossible to do justice to this substantial body of work, involving both algorithmic and game-theoretic ideas; for a start, see the surveys [19, 12].

1.2 Significance and Practical Impact

Google's AdWords marketplace generates multi-billion dollar revenues annually and the current annual worldwide spending on digital advertising is almost half a trillion dollars. These revenues of Google and other Internet services companies enable them to offer crucial services, such as search, email, videos, news, apps, maps etc. for free – services that have virtually transformed our lives.

We note that SMALL is the most relevant case of adwords for the search ads marketplace e.g., see [6]. A remarkable feature of Google, and other search engines, is the speed with which they are able to show search results, often in milliseconds. In order to show ads at the same speed, together with search results, the solution for SMALL needed to be minimalistic in its use of computing power, memory and communication.

The MSVV Algorithm satisfied these criteria and therefore had substantial impact in this marketplace. Furthermore, the idea underlying their algorithm was extracted into a simple heuristic, called *bid scaling*, which uses even less computation and is widely used by search engine companies today. As mentioned above, our Conditional Algorithm for SMALL is even more elementary and is budget-oblivious.

It will be useful to view the AdWords marketplace in the context of a bigger revolution, namely the advent of the Internet and mobile computing, and the consequent resurgence of the area of matching-based market design. The birth of this area goes back to the seminal

1962 paper of Gale and Shapley on stable matching [10]. Over the decades, this area became known for its highly successful applications, having economic as well as sociological impact. These included matching medical interns to hospitals, students to schools in large cities, and kidney exchange.

The resurgence led to a host of highly innovative and impactful applications. Besides the AdWords marketplace, which matches queries to advertisers, these include Uber, matching drivers to riders; Upwork, matching employers to workers; and Tinder, matching people to each other, see [14, 8] for more details.

A successful launch of such markets calls for economic and game-theoretic insights, together with algorithmic ideas. The Gale-Shapley Deferred Acceptance Algorithm and its follow-up works provided the algorithmic backbone for the "first life" of matching-based market design. The algorithm Ranking has become the paradigm-setting algorithmic idea in the "second life" of this area [8]. Interestingly enough, this result was obtained in the pre-Internet days, over thirty years ago.

1.3 Technical Ideas

Our extension from Ranking to Small needs to go via Adwords. It turns out that Adwords suffers from an inherent structural difficulty, see the full paper [24]. We temporarily finesse this difficulty by using the idea of "fake" money. The expected revenue of our online algorithm for Adwords is at least $(1 - 1/e)$ fraction of the optimal offline revenue; however, this total revenue consists of real as well as fake money. We provide an upper-bound on the fake money in the worst case, and this suffices to show that, asymptotically, the fake money used by Small, is negligible. Determining the true competitive ratio of our algorithm for Adwords is left as an interesting and important open problem.

As described in Section 1.1, Single-Valued can be reduced to online vertex weighted matching, by making k_j copies of each advertiser j; however, this reduction does not work for Adwords. The reason is that the manner in which budget B_j of bidder j gets partitioned into bids is not predictable in the latter problem; it depends on the queries, their order of arrival and the randomization executed in a run of the algorithm. Therefore, in order to build techniques to attack Adwords, we will first need to solve Single-Valued *without* reducing it to online vertex weighted matching.

This is done in Algorithm 1. Almost all of our new ideas, on the probabilistic front, needed to attack Small were obtained in the process analyzing this algorithm. First, since vertex j is not split into k_j copies, we cannot talk about the contribution of edges anymore. Even worse, we don't have individual vertices for keeping track of the revenue accrued from each match, as per the scheme of [9]. Our algorithm gets around this difficulty by accumulating revenue in the same "account" each time bidder j gets matched. The corresponding random variable, r_j, is called the *total revenue* of bidder j, for want of a better name, see Remark 4. Lower bounding $\mathbb{E}[r_j]$ is much more tricky than lower bounding the revenue of a good in OBM, since it involves "teasing apart" the k_j accumulations made into this account; this is done in Lemma 14.

The key fact needed in the analysis of Ranking is that for each edge $e = (i, j)$ in the underlying graph, its expected contribution to the matching produced is at least $(1 - 1/e)$. For this purpose, the random variable, u_e, called *threshold*, is defined in [23].

For analyzing Single-Valued, a replacement is needed for this lemma. For this purpose, we give the notion of a *j-star*, denoted X_j, which consists of bidder j together with edges to k_j of its neighbors in G, see Definition 10. The contribution of j-star X_j, is denoted by $\mathbb{E}[X_j]$, which is also defined in Definition 10. Finally, using the lower bound on $\mathbb{E}[r_j]$, Lemma 14 gives a lower $\mathbb{E}[X_j]$ for *every* j-star, X_j. This lemma crucially uses a new random variable, called *truncated threshold*, see Definition 9.

Next, we explain the reason for truncation in the definition of this random variable. Consider bidder j and a query i_l that is desired by j. Observe that in run $\mathcal{R}_j$[7], query i_l can get a bid as large as $B \cdot (1 - \frac{1}{e})$, where $B = \max_{k \in A}\{b_k\}$, whereas the largest bid that j can make to i_l is $b_j \cdot (1 - \frac{1}{e})$; in general, b_j may be smaller than B. Now, i_l contributes revenue to r_j only if i_l is matched to j in run $\mathcal{R}$, an event which will definitely not happen if $u_{e_l} > b_j \cdot (1 - \frac{1}{e})$. Therefore, whenever $u_{e_l} \in [b_j \cdot (1 - \frac{1}{e}),\ B \cdot (1 - \frac{1}{e})]$, the contribution to r_j is zero. By truncating u_{e_l} to $b_j \cdot (1 - \frac{1}{e})$, we have effectively changed the probability density function of u_{e_l} so that the probability of the event $u_{e_l} \in [b_j \cdot (1 - \frac{1}{e}), B \cdot (1 - \frac{1}{e})]$ is now concentrated at the event $u_{e_l} = b_j \cdot (1 - \frac{1}{e})$. From the viewpoint of lower bounding the revenue accrued in r_j, the two probability density functions are equivalent since the revenue accrued is zero under both these events. On the other hand, the truncated random variable enables us to apply the law of total expectation, in the proof of Lemma 14, in the same way as it was done in [23], without introducing more difficulties.

Finally, in order to establish the no-surpassing property for Single-Valued, we give the necessary combinatorial facts in Lemma 7 and Corollary 8. These facts are enhanced versions of the facts needed to prove the no-surpassing property for Ranking in [23].

2 Preliminaries

Online Bipartite Matching (OBM). Let B be a set of n buyers and S a set of n goods. A bipartite graph $G = (B, S, E)$ is specified on vertex sets B and S, and edge set E, where for $i \in B,\ j \in S$, $(i, j) \in E$ if and only if buyer i *likes* good j. G is assumed to have a perfect matching and therefore each buyer can be given a unique good she likes. Graph G is revealed in the following manner. The n goods are known up-front. On the other hand, the buyers arrive one at a time, and when buyer i arrives, the edges incident at i are revealed.

We are required to design an online algorithm $\mathcal{A}$ in the following sense. At the moment a buyer i arrives, the algorithm needs to match i to one of its unmatched neighbors, if any; if all of i's neighbors are matched, i remains unmatched. The difficulty is that the algorithm does not "know" the edges incident at buyers which will arrive in the future and yet the size of the matching produced by the algorithm will be compared to the best *off-line matching*; the latter of course is a perfect matching. The formal measure for the algorithm is defined in Section 2.1.

Adwords Problem (ADWORDS). Let A be a set of m *advertisers*, also called *bidders*, and Q be a set of n *queries*. A bipartite graph $G = (Q, A, E)$ is specified on vertex sets Q and A, and edge set E, where for $i \in Q$ and $j \in A$, $(i, j) \in E$ if and only if bidder j is *interested in* query i. Each query i needs to be matched[8] to at most one bidder who is interested in it. For each edge (i, j), bidder j *knows* his bid for i, denoted by $\text{bid}(i, j) \in \mathbb{Z}_+$. Each bidder also has a *budget* $B_j \in \mathbb{Z}_+$ which satisfies $B_j \geq \text{bid}(i, j)$, for each edge (i, j) incident at j.

Graph G is revealed in the following manner. The m bidders are known up-front and the queries arrive one at a time. When query i arrives, the edges incident at i are revealed, together with the bids associated with these edges. If i gets matched to j, then the matched edge (i, j) is assigned a weight of $\text{bid}(i, j)$. The constraint on j is that the total weight of matched edges incident at it be at most B_j. The objective is to maximize the total weight of all matched edges at all bidders.

---

[7] Run $\mathcal{R}_j$ is defined in Definition 6.

[8] Clearly, this is not a matching in the usual sense, since a bidder may be matched to several queries.

Adwords under Single-Valued Bidders (SINGLE-VALUED). SINGLE-VALUED is a special case of ADWORDS in which each bidder j will make bids of a single value, $b_j \in \mathbb{Z}_+$, for the queries he is interested in. If i accepts j's bid, then i will be matched to j and the weight of this matched edge will be b_j. Corresponding to each bidder j, we are also given $k_j \in \mathbb{Z}_+$, the maximum number of times j can be matched to queries. The objective is to maximize the total weight of matched edges. Observe that the matching M found in G is a b-matching with the b-value of each query i being 1 and of advertiser j being k_j.

Adwords under Small Bids (SMALL). SMALL is a special case of ADWORDS in which for each bidder j, each bid of j is small compared to its budget. Formally, we will capture this condition by imposing the following constraint. For a valid instance I of SMALL, define

$$\mu(I) = \max_{j \in A} \left\{ \frac{\max_{(i,j)\in E} \{\text{bid}(i,j) - 1\}}{B_j} \right\}.$$

Then we require that

$$\lim_{n(I)\to\infty} \mu(I) = 0,$$

where $n(I)$ denotes the number of queries in instance I.

2.1 The competitive ratio of online algorithms

We will define the notion of competitive ratio of a randomized online algorithm in the context of OBM.

▶ **Definition 2.** *Let $G = (B, S, E)$ be a bipartite graph as specified above. The competitive ratio of a randomized algorithm $\mathcal{A}$ for OBM is defined to be:*

$$c(\mathcal{A}) = \min_{G=(B,S,E)} \min_{\rho(B)} \frac{\mathbb{E}[\mathcal{A}(G, \rho(B))]}{n},$$

where $\mathbb{E}[\mathcal{A}(G, \rho(B))]$ is the expected size of matching produced by $\mathcal{A}$; the expectation is over the random bits used by $\mathcal{A}$. We may assume that the worst case graph and the order of arrival of buyers, given by $\rho(B)$, are chosen by an adversary who knows the algorithm. It is important to note that the algorithm is provided random bits after *the adversary makes its choices.*

▶ Remark 3. For each problem studied in this paper, we will assume that the offline matching is complete. It is easy to extend the arguments, without changing the competitive ratio, in case the offline matching is not complete.

3 Algorithm for Single-Valued

Algorithm 1, which will be denoted by $\mathcal{A}_1$, is an online algorithm for SINGLE-VALUED. Before execution of Step (1) of $\mathcal{A}_1$, the order of arrival of queries, say $\rho(B)$, is fixed by the adversary. We will define several random variables whose purpose will be quite similar to that in RANKING and they will be given similar names as well; however, their function is not as closely tied to these economics-motivated names as in RANKING, see also Remark 4. Three of these random variables are the *price* p_j and *total revenue* r_j of each bidder $j \in A$, and the *utility* u_i of each query $i \in Q$.

We now describe how values are assigned to these random variables in a run of Algorithm 1. In Step (1), for each bidder j, $\mathcal{A}_1$ picks a price $p_j \in [\frac{1}{e}, 1]$ via the specified randomized process. Furthermore, the revenue r_j and *degree* d_j of bidder j are both initialized to zero, the latter represents the number of times j has been matched. During the run of $\mathcal{A}_1$, j will get matched to at most k_j queries; each match will add b_j to the total revenue generated by the algorithm. b_j is broken into a revenue and a utility component, with the former being added to r_j and the latter forming u_i. At the end of $\mathcal{A}_1$, r_j will contain all the revenue accrued by j.

In Step (2), on the arrival of query i, we will say that bidder j is *available* if $(i, j) \in E$ and $d_j < k_j$. At this point, for each available bidder j, the *effective bid* of j for i is defined to be $\text{ebid}(j) = b_j \cdot (1 - p_j)$; clearly, $\text{ebid}(j) \in [0, b_j \cdot \left(1 - \frac{1}{e}\right)]$. Query i accepts the bidder whose effective bid is the largest. If there are no bids, matching M remains unchanged. If i accepts j's bid, then edge (i, j) is added to matching M and the weight of this edge is set to b_j. Furthermore, the *utility* of i, u_i, is defined to be $\text{ebid}(j)$ and the revenue r_j of j is incremented by $b_j \cdot p_j$. Once all queries are processed, matching M and its weight W are output.

▶ Remark 4. [9] had given the economics-based names of random variables for their proof of Ranking. Although we have used the same names for similar random variables in Section 3 for Single-Valued, the reader should not attribute an economic interpretation to these the names[9].

3.1 Analysis of Algorithm 1

For the analysis of Algorithm $\mathcal{A}_1$, we will use the random variables W, p_j, r_j and u_i defined above; their values are fixed during the execution of $\mathcal{A}_2$. In addition, corresponding to each edge $e = (i, j) \in E$, in Definition 9, we will introduce a new random variable, u_e, which will play a central role.

▶ **Lemma 5.**

$$\mathbb{E}[W] \;= \sum_{i}^{n} \mathbb{E}\left[u_i\right] \;+\; \sum_{j}^{m} \mathbb{E}[r_j].$$

Proof. For each edge $(i, j) \in M$, its contribution to W is b_j. Furthermore, the sum of u_i and the contribution of (i, j) to r_j is also b_j. This gives the first equality below. The second equality follows from linearity of expectation.

$$\mathbb{E}[W] \;= \mathbb{E}\left[\sum_{i=1}^{n} u_i \;+\; \sum_{j=1}^{m} r_j\right] \;= \sum_{i}^{n} \mathbb{E}\left[u_i\right] \;+\; \sum_{j}^{m} \mathbb{E}[r_j],$$ ◀

▶ **Definition 6.** *We will define several runs of Algorithm 1. In these runs, we will assume Step (1) is executed once. We next define several ways of executing Step (2). Let $\mathcal{R}$ denote the run of Step (2) on the entire graph G. Corresponding to each bidder $j \in A$, let G_j denote graph G with bidder j removed. Define $\mathcal{R}_j$ to be the run of Step (2) on graph G_j.*

[9] We failed to come up with more meaningful names for these random variables and therefore have stuck to the old names.

Algorithm 1 $\mathcal{A}_1$: Algorithm for SINGLE-VALUED.

1. **Initialization:** $M \leftarrow \emptyset$.
   $\forall j \in A$, do:
   a. Pick w_j uniformly from $[0,1]$ and set price $p_j \leftarrow e^{w_j - 1}$.
   b. $r_j \leftarrow 0$.
   c. $d_j \leftarrow 0$.

2. **Query arrival:** When query i arrives, **do**:
   a. $\forall j \in A$ s.t. $(i,j) \in E$ and $d_j < k_j$ **do**:
      i. ebid$(j) \leftarrow b_j \cdot (1 - p_j)$.
      ii. Offer effective bid of ebid(j) to i.
   b. Query i accepts the bidder whose effective bid is the largest.
   (If there are no bids, matching M remains unchanged.)
   If i accepts j's bid, then **do**:
      i. Set utility: $u_i \leftarrow b_j \cdot (1 - p_j)$.
      ii. Update revenue: $r_j \leftarrow r_j + b_j \cdot p_j$.
      iii. Update degree: $d_j \leftarrow d_j + 1$.
      iv. Update matching: $M \leftarrow M \cup (i,j)$. Define the weight of (i,j) to be b_j.
   c. **Output:** Output matching M and its total weight W.

Lemma 7 and Corollary 8 given below establish a relationship between the available bidders for a query i in the two runs $\mathcal{R}$ and $\mathcal{R}_j$. Note that bidders are available in multiplicity and therefore we will have to use the notion of a multiset rather than a set, as was done in [23].

A *multiset* contains elements with multiplicity. Let A and B be two multisets over n elements $\{1, 2, \ldots n\}$, and let $a_i \geq 0$ and $b_i \geq 0$ denote the multiplicities of element i in A and B, respectively. We will say that $A \subseteq B$ if for each i, $a_i \leq b_i$, and $A = B$ if for each i, $a_i = b_i$. We will say that $i \in A$ if $a_i \geq 1$. We will define $A \cap B$ to be the multiset containing each element i exactly $\min\{a_i, b_i\}$ times, and $A - B$ to be the multiset containing each element i exactly $\max\{a_i - b_i, 0\}$ times.

As before, let us renumber the queries so their order of arrival under $\rho(B)$ is $1, 2, \ldots n$. Let $T(i)$ and $T_j(i)$ denote the multisets of available bidders at the time of arrival of query i (i.e., just before the query i gets matched) in runs $\mathcal{R}$ and $\mathcal{R}_j$, respectively. In particular, $T(1)$ will contain k_l copies of l for each bidder l and $T_j(1)$ will contain k_l copies of l for each bidder l, other than j. Similarly, let $S(i)$ and $S_j(i)$ denote the projections of $T(i)$ and $T_j(i)$ on the neighbors of i in G and G_j, respectively.

We have assumed that Step (1) of Algorithm 1 has already been executed and a price p_k has been assigned to each bidder k. The effective bid of bidder k is ebid$(k) = b_k \cdot (1 - p_k)$. With probability 1, the effective bids of all bidders are distinct. Let F_1 be the multiset containing k_l copies of l for each $l \in A$ such that $b_l \cdot (1 - p_l) > b_j \cdot (1 - p_j)$. Similarly, let F_2 be the multiset containing k_l copies of l for each $l \in A$ such that and $b_l \cdot (1 - p_l) < b_j \cdot (1 - p_j)$. Observe that j is not contained in either multiset.

▶ **Lemma 7.** *For each i, $1 \leq i \leq n$, the following hold:*
1. $(T_j(i) \cap F_1) = (T(i) \cap F_1)$.
2. $(T_j(i) \cap F_2) \subseteq (T(i) \cap F_2)$.

Proof.

1. Clearly, in both runs, $\mathcal{R}$ and $\mathcal{R}_j$, any query having an available bidder in F_1 will match to the most profitable one of these, without even considering the rest of the bidders. Since $j \notin F_1$, the two runs behave in an identical manner on the set F_1, thereby proving the first statement.
2. The proof is by induction on i. The base case is trivially true because $(T_j(1) \cap F_2) = (T(1) \cap F_2)$, since $j \notin F_2$. Assume that the statement is true for $i = k$ and let us prove it for $i = k + 1$. By the first statement, we need to consider only the case that there are no available bidders for the k^{th} query in F_1 in the runs $\mathcal{R}$ and $\mathcal{R}_j$. Assume that in run $\mathcal{R}_j$, this query gets matched to bidder l; if it remains unmatched, we will take l to be null. Clearly, l is the most profitable bidder it is incident to in $T_j(k)$. Therefore, the most profitable bidder it is incident to in run $\mathcal{R}$ is the best of l, the most profitable bidder in $T(k) - T_j(k)$, and j, in case it is available. In each of these cases, the induction step holds. ◀

In the corollary below, the first two statements follow from Lemma 7 and the third statement follows from the first two statements.

▶ **Corollary 8.** *For each i, $1 \leq i \leq n$, the following hold:*

1. $(S_j(i) \cap F_1) = (S(i) \cap F_1)$.
2. $(S_j(i) \cap F_2) \subseteq (S(i) \cap F_2)$.
3. $S_j(i) \subseteq S(i)$.

Next we define a new random variable, u_e, for each edge $e = (i, j) \in E$. This is called the *truncated threshold* for edge e and is given in Definition 9. It is critically used in the proofs of Lemmas 13 and 14.

▶ **Definition 9.** *Let $e = (i, j) \in E$ be an arbitrary edge in G. Define random variable, u_e, called the* truncated threshold *for edge e, to be $u_e = \min\{ut_i,\ b_j \cdot \left(1 - \frac{1}{e}\right)\}$, where ut_i is the utility of query i in run $\mathcal{R}_j$.*

▶ **Definition 10.** *Let $j \in A$. Henceforth, we will denote k_j by k in order to avoid triple subscripts. Let $i_1, \ldots, i_k$ be queries such that for $1 \leq l \leq k$, $(i_l, j) \in E$. Then $(j;\ i_1, \ldots, i_k)$ is called a* j-star. *Let X_j denote this j-star. The contribution of X_j to $\mathbb{E}[W]$ is $\mathbb{E}[r_j] + \sum_{l=1}^{k} \mathbb{E}[u_{i_l}]$, and it will be denote by $\mathbb{E}[X_j]$.*

Corresponding to j-star $X_j = (j;\ i_1, \ldots, i_k)$, denote by e_l the edge $(i_l, j) \in E$, for $1 \leq l \leq k$. Furthermore, let u_{e_l} denote the truncated threshold random variable corresponding to e_l.

▶ **Property 11** (No-Surpassing for Single-Valued)**.** *Assume that Step 1 of Algorithm 1 has been executed and a price p_k has been assigned to each advertiser k. Suppose that the effective bid which query i gets in run $\mathcal{R}_j$ is less than $b_j \cdot (1 - p_j)$; the latter is clearly the effective bid which j makes to i in run $\mathcal{R}$. Then, in run $\mathcal{R}$, no bid to i will surpass* $\text{ebid}(j) = b_j \cdot (1 - p_j)$.

▶ **Lemma 12.** *The No-Surpassing Property holds for Single-Valued.*

Proof. Suppose the bid of j, namely $b_j \cdot (1 - p_j)$, is better than the best bid that buyer i gets in run $\mathcal{R}_j$. If so, i gets no bid from F_1 in $\mathcal{R}_j$; observe that they are all higher than $b_j \cdot (1 - p_j)$. Now, by the first part of Corollary 8, i gets no bid from F_1 in run $\mathcal{R}$ as well, i.e., in run $\mathcal{R}$, no bid to i will surpass $b_j \cdot (1 - p_j)$. ◀

▶ **Lemma 13.** *Corresponding to j-star $X_j = (j;\ i_1, \ldots, i_k)$, the following hold.*
For $1 \leq l \leq k$, $u_{i_l} \geq u_{e_l}$.

Proof. By the third statement of Corollary 8, i_l has more options in run $\mathcal{R}$ as compared to run $\mathcal{R}_j$. Furthermore, the truncation of the random variable only aids the inequality needed and therefore $u_{i_l} \geq u_{e_l}$. ◀

Our next goal is to lower bound the contribution of an arbitrary j-star, $\mathbb{E}[X_j]$, which in turn involves lower bounding $\mathbb{E}[r_j]$. The latter crucially uses the fact that p_j is independent of u_{e_l}. This follows from the fact that u_{e_l} is determined by run $\mathcal{R}_j$ on graph G_j, which does not contain vertex j.

▶ **Lemma 14.** *Let $j \in A$ and let $X_j = (j;\ i_1, \ldots, i_k)$ be a j-star. Then*

$$\mathbb{E}[X_j] \ \geq\ k \cdot b_j \cdot \left(1 - \frac{1}{e}\right).$$

Proof. We will first lower bound $\mathbb{E}[r_j]$. Let $f_U(b_j \cdot z_1, \ldots b_j \cdot z_k)$ be the joint probability density function of $(u_{e_1}, \ldots u_{e_k})$; clearly, $f_U(b_j \cdot z_1, \ldots b_j \cdot z_k)$ can be non-zero only if $z_l \in [0, 1 - \frac{1}{e}]$, for $1 \leq l \leq k$. By the law of total expectation,

$$\mathbb{E}[r_j] \ = \int_{(z_1,\ldots,z_k)} \mathbb{E}[r_j \mid u_{e_1} = b_j \cdot z_1, \ldots, u_{e_k} = b_j \cdot z_k] \cdot f_U(b_j \cdot z_1, \ldots b_j \cdot z_k)\ dz_1 \ldots dz_k,$$

where the integral is over $z_l \in [0, \left(1 - \frac{1}{e}\right)]$, for $1 \leq l \leq k$.

For lower-bounding the conditional expectation in this integral, let $w_l \in [0, 1]$ be s.t. $e^{w_l - 1} = 1 - z_l$, for $1 \leq l \leq k$. For $x \in [0, 1]$, define the set $S(x) = \{l \mid 1 \leq l \leq k \text{ and } x < w_l\}$.

▷ Claim 15. Conditioned on $(u_{e_1} = b_j \cdot z_1, \ldots, u_{e_k} = b_j \cdot z_k)$, if $p_j = e^{x-1}$, then the degree of j at the end of Algorithm $\mathcal{A}_2$ is at least $|S(x)|$, i.e., the contribution to r_j in this run was $\geq b_j \cdot p_j \cdot |S(x)|$.

Proof. Suppose $l \in S(x)$, then $x < w_l$. In run $\mathcal{R}_j$, the maximum effective bid that i_l received has value $b_j \cdot z_l$. In run $\mathcal{R}$, if at the arrival of query i_l, j is already fully matched, the contribution to r_j in this run was $k \cdot b_j \cdot p_j$ and the claim is obviously true. If not, then since $x < w_l$, $b_j \cdot (1 - p_j) > b_j \cdot z_l$. The crux of the matter is that by Lemma 12, the No-Surpassing Property holds. Therefore, query i_l will receive its largest effective bid from j, i_l will get matched to it, and r_j will be incremented by $b_j \cdot p_j$. The claim follows. ◁

For $1 \leq l \leq k$, define indicator functions $I_l : [0, 1] \to \{0, 1\}$ as follows.

$$I_l(x) \ = \begin{cases} 1 & \text{if } x < w_l, \\ 0 & \text{otherwise.} \end{cases}$$

Clearly, $|S(x)| = \sum_{l=1}^{k} I_j(x)$. By Claim 15,

$$\mathbb{E}[r_j \mid u_{e_1} = b_j \cdot z_1, \ldots, u_{e_k} = b_j \cdot z_k] \ \geq\ b_j \cdot \int_0^1 |S(x)| \cdot e^{x-1}\ dx$$

$$= \ b_j \cdot \int_0^1 \sum_{l=1}^{k} I_l(x) \cdot e^{x-1}\ dx \ = \ b_j \cdot \sum_{l=1}^{k} \int_0^1 I_l(x) \cdot e^{x-1}\ dx \ = \ b_j \cdot \sum_{l=1}^{k} \int_0^{w_l} e^{x-1}\ dx$$

$$= \ b_j \cdot \sum_{l=1}^{k} \left(e^{w_l - 1} - \frac{1}{e}\right) \ = \ b_j \cdot \sum_{l=1}^{k} \left(1 - \frac{1}{e} - z_l\right).$$

Since $I_l(x) = 0$ for $x \in [w_l, 1]$, we get that $\int_0^1 I_l(x) \cdot e^{x-1}\, dx = \int_0^{w_l} e^{x-1}\, dx$; this fact has been used above. Therefore,

$$\mathbb{E}[r_j] = \int_{(z_1,\ldots,z_k)} \mathbb{E}[r_j \mid u_{e_1} = b_j \cdot z_1, \ldots, u_{e_k} = b_j \cdot z_k] \cdot f_U(b_j \cdot z_1, \ldots b_j \cdot z_k)\, dz_1 \ldots dz_k$$

$$\geq b_j \cdot \int_{(z_1,\ldots,z_k)} \sum_{l=1}^{k} \left(1 - \frac{1}{e} - z_l\right) \cdot f_U(b_j \cdot z_1, \ldots b_j \cdot z_k)\, dz_1 \ldots dz_k$$

$$= k \cdot b_j \cdot \left(1 - \frac{1}{e}\right) - \sum_{l=1}^{k} \mathbb{E}[u_{e_l}],$$

where both integrals are over $z_l \in [0, \left(1 - \frac{1}{e}\right)]$, for $1 \leq l \leq k$.

By Lemma 13, $\mathbb{E}[u_{i_l}] \geq \mathbb{E}[u_{e_l}]$, for $1 \leq l \leq k$. Hence we get

$$\mathbb{E}[X_j] = \mathbb{E}[r_j] + \sum_{l=1}^{k} \mathbb{E}[u_{i_l}] \geq k \cdot b_j \cdot \left(1 - \frac{1}{e}\right), \qquad \blacktriangleleft$$

▶ **Lemma 16.**

$$\mathbb{E}[W] = \sum_{i}^{n} \mathbb{E}[u_i] + \sum_{j}^{m} \mathbb{E}[r_j].$$

Proof. By definition of the random variables,

$$\mathbb{E}[W] = \mathbb{E}\left[\sum_{i=1}^{n} u_i + \sum_{j=1}^{m} r_j\right] = \sum_{i}^{n} \mathbb{E}[u_i] + \sum_{j}^{m} \mathbb{E}[r_j],$$

where the first equality follows from the fact that if $(i, j) \in M$ then W is incremented by b_j and $u_i + r_j = b_j$. The second equality follows from linearity of expectation. ◀

▶ **Theorem 17.** *The competitive ratio of Algorithm $\mathcal{A}_2$ is at least $1 - \frac{1}{e}$. Furthermore, it is budget-oblivious.*

Proof. Let P denote a maximum weight b-matching in G, computed in an offline manner. By the assumption made in Remark 3, its weight is

$$w(P) = \sum_{j=1}^{m} k_j \cdot b_j.$$

Let T_j denote the j-star, under P, corresponding to each $j \in A$. The expected weight of matching produced by $\mathcal{A}_2$ is

$$\mathbb{E}[W] = \sum_{i=1}^{n} \mathbb{E}[u_i] + \sum_{j=1}^{m} \mathbb{E}[r_j] = \sum_{j=1}^{m} \mathbb{E}[T_j] \geq \sum_{j=1}^{m} b_j \cdot k_j \left(1 - \frac{1}{e}\right) = \left(1 - \frac{1}{e}\right) \cdot w(P),$$

where the first equality uses Lemma 16, the second follows from linearity of expectation and the inequality follows from Lemma 14.

Finally, Algorithm $\mathcal{A}_2$ is budget-oblivious because it does not need to know k_j for bidders j; it only needs to know during a run whether the k_j bids available to bidder j have been exhausted. The theorem follows. ◀

References

1 Gagan Aggarwal, Ashwinkumar Badanidiyuru, and Aranyak Mehta. Autobidding with constraints. In *International Conference on Web and Internet Economics*, pages 17–30. Springer, 2019.

2 Gagan Aggarwal, Gagan Goel, Chinmay Karande, and Aranyak Mehta. Online vertex-weighted bipartite matching and single-bid budgeted allocations. In *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms*, pages 1253–1264, 2011.

3 Susanne Albers and Sebastian Schubert. Optimal algorithms for online b-matching with variable vertex capacities. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021.

4 Benjamin Birnbaum and Claire Mathieu. On-line bipartite matching made simple. *ACM Sigact News*, 39(1):80–87, 2008.

5 Niv Buchbinder, Kamal Jain, and Joseph Seffi Naor. Online primal-dual algorithms for maximizing ad-auctions revenue. In *European Symposium on Algorithms*, pages 253–264, 2007.

6 Nikhil Devanur and Aranyak Mehta. Online matching in advertisement auctions. In Federico Echenique, Nicole Immorlica, and Vijay V. Vazirani, editors, *Online and Matching-Based Market Design*. Cambridge University Press, 2022. [To appear] `https://www.ics.uci.edu/~vazirani/AdAuctions.pdf`.

7 Nikhil R Devanur, Kamal Jain, and Robert D Kleinberg. Randomized primal-dual analysis of ranking for online bipartite matching. In *Proceedings of the twenty-fourth annual ACM-SIAM symposium on Discrete algorithms*, pages 101–107. SIAM, 2013.

8 Federico Echenique, Nicole Immorlica, and Vijay V. Vazirani, editors. *Online and Matching-Based Market Design*. Cambridge University Press, 2023.

9 Alon Eden, Michal Feldman, Amos Fiat, and Kineret Segal. An economic-based analysis of ranking for online bipartite matching. In *SIAM Symposium on Simplicity in Algorithms*, 2021.

10 David Gale and Lloyd S Shapley. College admissions and the stability of marriage. *The American Mathematical Monthly*, 69(1):9–15, 1962.

11 Gagan Goel and Aranyak Mehta. Online budgeted matching in random input models with applications to adwords. In *SODA*, volume 8, pages 982–991, 2008.

12 Zhiyi Huang and Thorben Trobst. Online matching. In Federico Echenique, Nicole Immorlica, and Vijay V. Vazirani, editors, *Online and Matching-Based Market Design*. Cambridge University Press, 2022. [To appear] `https://www.ics.uci.edu/~vazirani/Ch4.pdf`.

13 Zhiyi Huang, Qiankun Zhang, and Yuhao Zhang. Adwords in a panorama. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1416–1426. IEEE, 2020.

14 Simons Institute. Online and matching-based market design, 2019. URL: `https://simons.berkeley.edu/programs/market2019`.

15 Kamal Jain, Mohammad Mahdian, Evangelos Markakis, Amin Saberi, and Vijay V Vazirani. Greedy facility location algorithms analyzed using dual fitting with factor-revealing LP. *Journal of the ACM (JACM)*, 50(6):795–824, 2003.

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

17 Richard M Karp, Umesh V Vazirani, and Vijay V Vazirani. An optimal algorithm for on-line bipartite matching. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 352–358, 1990.

18 Jingxun Liang, Zhihao Gavin Tang, Yixuan Xu, Yuhao Zhang, and Renfei Zhou. On the perturbation function of ranking and balance for weighted online bipartite matching. *arXiv preprint*, 2022. `arXiv:2210.10370`.

19 Aranyak Mehta. *Online matching and ad allocation*, volume 8(4). Now Publishers, Inc., 2013.

20 Aranyak Mehta, Amin Saberi, Umesh Vazirani, and Vijay Vazirani. Adwords and generalized online matching. *Journal of the ACM (JACM)*, 54(5), 2007.

21 Rajan Udwani. Adwords with unknown budgets and beyond. *arXiv preprint*, 2021. `arXiv:2110.00504`.

22 Vijay V Vazirani. Online bipartite matching and adwords. *arXiv preprint*, 2021. `arXiv:2107.10777`.

23 Vijay V Vazirani. Online bipartite matching and adwords. In *47th International Symposium on Mathematical Foundations of Computer Science*, 2022.

24 Vijay V Vazirani. Towards a practical, budget-oblivious algorithm for the adwords problem under small bids. *arXiv preprint*, 2023. `arXiv:2107.10777`.