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

Vijay V. Vazirani[*1]

[1]University of California, Irvine

Abstract

Motivated by recent insights on 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 [MSVV07]. 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. Our algorithm also provides an avenue for improving the bound for the general adwords problem from 0.5016, given in [HZZ20].

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 [MSVV07]. If further experiments confirm this, our algorithm may be useful as such in practice, especially because of its budget-obliviousness.

[*]Supported in part by NSF grant CCF-1815901.

1 Introduction

The *adwords problem*, called General[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. The overall goal is to maximize the total revenue. This problem is notoriously difficult and has remained largely unsolved; see Section 1.1 for marginal progress made recently. Its special case, when bids are small compared to budgets, called Small, 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 [MSVV07]; for the impact of this result in the marketplace, see Section 1.2. We will call the algorithm given in [MSVV07] the MSVV Algorithm.

In Open Problem Number 20 in [Meh13], Mehta asks for a *budget-oblivious online algorithm* for Small. Such an algorithm does not know the daily budgets of advertisers; it only 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 platforms [ABM19, DM22], which manage the ad campaigns of large advertisers; they dynamically adjust bids and budgets over multiple search engines to improve performance. The greedy algorithm, which matches an arriving query to the advertiser making the highest bid, is clearly budget-oblivious; its competitive ratio is 0.5. An improved algorithm, having a competitive ratio of 0.522, was recently obtained by Udwani [Udw21], using the idea of an LP-free analysis, which involves writing appropriate linear inequalities to compare the online algorithm with the offline optimal algorithm.

General is a generalization of the online bipartite matching problem (OBM); for the latter, an optimal algorithm, called Ranking, achieving a competitive ratio of $\left(1 - \frac{1}{e}\right)$, was given in [KVV90]. However, the analysis of Ranking given in [KVV90] was difficult to comprehend. A sequence of papers has finally led to a simple and elegant analysis, see Section 1.1. This motivated us to extend Ranking all the way to Small, since the simplicity of Ranking, and the more basic nature of this approach to Small than the one used in [MSVV07] (see Section 1.1), led to the hope that it would yield an algorithm with better properties, e.g., budget-obliviousness.

However, since extending Ranking directly to Small seemed an uphill task, we decided to attempt 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 [AGKM11] 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 it directly and not resort to this reduction.

We give an optimal, budget-oblivious algorithm for Single-Valued. Furthermore, our algorithm uses fewer random bits than the previous approach of [AGKM11]; see Section 1.1 for a detailed comparison. We note that in contemporary[2] and independent work, Albers and Schubert [AS21] 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

[1]For formal statements of problems studied in this paper, see Section 2
[2]Our paper was first posted on arXiv on July 22, 2021 [Vaz21].

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 10. 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 [MSVV07]. As can be seen in the four Tables in Section 4.5, 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.

Our algorithms for Single-Valued and Small are direct generalizations of Ranking and retain its simplicity. Since the ideas underlying our algorithm and the probabilistic part of the proof for Small are quite substantial, we have stated them formally, after assuming the No-Surpassing Property, see Section 4.2. We leave the open problem of removing this assumption. Another viewpoint is that because of its budget-obliviousness, the algorithm may be useful as such in practice, especially if further experiments confirm that its performance is good.

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 [KVV90] 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 [GM08, BM08], got the ball rolling, setting the stage for the substantial simplification given in [DJK13], using a randomized primal-dual approach. [DJK13] 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[3].

Interestingly enough, the next simplification involved removing the scaffolding of LP-duality and casting the proof in purely probabilistic terms[4], 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 [EFFS21]. We note that when we move to generalizations of OBM,

[3]For a succinct proof of optimality of the underlying function, e^{x-1}, see Section 2.1.1 in [HT22].

[4]Even though there is no overt use of LP-duality in the proof of [EFFS21], it is unclear if this proof could have been obtained directly, without going the LP-duality-route.

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 [Vaz22].

An important generalization of OBM is online b-matching. This problem is a special case of General in which the budget of each advertiser is \$$b$ and the bids are 0/1. [KP00] 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. [KP00] 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[5] as follows: [MSVV07] first gave a simpler proof of the competitive ratio of BALANCE using the notion of a *factor-revealing LP* [JMM$^+$03]. Then they gave the notion of a *tradeoff-revealing LP*, which yielded an algorithm achieving a competitive ratio of $\left(1 - \frac{1}{e}\right)$. [MSVV07] 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, [KP00] 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 [MSVV07], a second optimal online algorithm for Small was given in [BJN07], 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. [AGKM11] extended Ranking to obtain an optimal online algorithm for this problem. Clearly, Single-Valued is intermediate between General and online vertex weighted matching. [AGKM11] 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 [AS21], needs to use only $|A|$ numbers.

As stated in the Introduction, General is a notoriously difficult problem, partly due to its inherent structural difficulties, which are described in Section 4.1. For General, 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 [HZZ20] 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 [MSVV07], search engine companies generously invested in research on models derived from OBM and adwords. The reason was two-fold: the substantial impact of [MSVV07] 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 [Meh13, HT22].

[5]It is worth recalling that [MSVV07] had first attempted extending OBM to Small; however, in the absence of new insights into OBM, this did not go very far.

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 [DM22]. 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 [GS62]. 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 [Ins19] 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. 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 GENERAL. As stated in the Introduction, GENERAL suffers from an inherent structural difficulty, see Section 4.1. We temporarily finesse this difficulty by using the idea of "fake" money. The expected revenue of our online algorithm for GENERAL 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 General is left as an interesting and important open problem, see Section 5.

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 General. 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 General, 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 [EFFS21]. 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 13.

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 [Vaz22].

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 9. The contribution of j-star X_j, is denoted by $\mathbb{E}[X_j]$, which is also defined in Definition 9. Finally, using the lower bound on $\mathbb{E}[r_j]$, Lemma 13 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 8.

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 $\mathcal{R}_j$, 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 13, in the same way as it was done in [Vaz22], without introducing more difficulties.

Finally, in order to establish the no-surpassing property for Single-Valued, we give the necessary combinatorial facts in Lemma 6 and Corollary 7. These facts are enhanced versions of the facts needed to prove the no-surpassing property for Ranking in [Vaz22].

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.

General Adwords Problem (General): 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[6] to at most one bidder who is interested in it. For each edge (i, j), bidder j *knows* his bid for i, denoted by $\mathrm{bid}(i, j) \in \mathbb{Z}_+$. Each bidder also has a *budget* $B_j \in \mathbb{Z}_+$ which satisfies $B_j \geq \mathrm{bid}(i, j)$, 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 $\mathrm{bid}(i, j)$. 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.

Adwords under Single-Valued Bidders (Single-Valued): Single-Valued is a special case of General 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 General 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

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

Then we require that

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

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

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. As an example, this is done for OBM in Remark **??**.

3 Algorithm for Single-Valued

Algorithm 1, which will be denoted by $\mathcal{A}_2$, is an online algorithm for Single-Valued. Before execution of Step (1) of $\mathcal{A}_2$, 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, $\mathcal{A}_2$ 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 $\mathcal{A}_2$, 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 $\mathcal{A}_2$, 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, $\text{ebid}(j) \in [0, b_j \cdot (1 - \frac{1}{e})]$. 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. [EFFS21] 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 Sections 3 and 4.2, for SINGLE-VALUED and GENERAL, the reader should not attribute an economic interpretation to these the names[7].

3.1 Analysis of Algorithm 1

For the analysis of Algorithm $\mathcal{A}_2$, 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 8, 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.

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

□

Next, 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.

Lemma 6 and Corollary 7 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 [Vaz22].

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

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

Algorithm 1. **($\mathcal{A}_2$: 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 $W$.
```

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 6. *For each i, $1 \le i \le n$, 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 6 and the third statement follows from the first two statements.

Corollary 7. *For each i, $1 \le i \le n$, 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 8. It is critically used in the proofs of Lemmas 12 and 13.

Definition 8. 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 $u_e = \min\{ut_i,\ b_j \cdot (1 - \frac{1}{e})\}$, where ut_i is the utility of query i in run $\mathcal{R}_j$.

Definition 9. 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 $1 \le l \le k$, $(i_l, j) \in E$. 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 $1 \le l \le k$. Furthermore, let u_{e_l} denote the truncated threshold random variable corresponding to e_l.

Property 10. (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 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 11. *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 7, 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 12. *Corresponding to j-star $X_j = (j;\ i_1, \ldots, i_k)$, the following hold.*

- *For $1 \le l \le k$, $u_{i_l} \ge u_{e_l}$.*

Proof. By the third statement of Corollary 7, 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 $u_{i_l} \geq u_{e_l}$. $\square$

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 13. *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 $z_l \in [0, (1 - \frac{1}{e})]$, 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 14. *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 11, 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. $\square$

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 14,

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

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

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

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

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

□

Lemma 15.

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

Proof. By definition of the random variables,

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

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 16. *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 15, the second follows from linearity of expectation and the inequality follows from Lemma 13.

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. $\square$

4 Algorithm for Small, After Assuming the No-Surpassing Property

Two new difficulties arise for the problem General The first is the inherent structural difficulty described in Section 4.1. Second, since bidders can have different bids for different queries, the no-surpassing property does not hold anymore, see Example 17.

Example 17. Assume that in the given instance for General, j, j' are two of the bidders, and $1, \ldots, k$ are k of the queries, where k is a large number. Assume $\text{bid}(l, j) = \alpha$ for $1 \leq l \leq k$ and $\text{bid}(l, j') = \alpha - 1$ for $1 \leq l \leq k - 1$. Further, assume that $\text{bid}(k, j') = (\alpha - 1) \cdot (k - 1)$. Let the budgets be $B_j = \alpha \cdot k$ and $B_{j'} = (\alpha - 1) \cdot (k - 1)$.

Now consider a run in which $p_j = p_{j'} = p$. Assume that in run $\mathcal{R}_j$, the best effective bid to $1, \ldots, k-1$ comes from j', and in run $\mathcal{R}_j$, the best effective bid to $1, \ldots, k-1$ comes from j. In run $\mathcal{R}_j$, the budget of j' is exhausted when k arrives and assume that k does not get any bids, making $u_e = 0$ for $e = (k, j)$. Now in run $\mathcal{R}$, $\text{ebid}(k, j) = \alpha(1 - p)$ and $\text{ebid}(k, j') = (\alpha - 1) \cdot (k - 1) \cdot (1 - p)$. Thus, even though $\text{ebid}(k, j) > u_e$, k will be matched to j' and not j. Clearly, this phenomenon will hold for all runs in which $p_{j'}$ is not too much larger than p_j.

For the rest of this section, we will make this assumption:

Assumption of No-Surpassing for General: The following holds:
Assume that Step 1 of Algorithm 2 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 $\text{bid}(i, j) \cdot (1 - p_j)$; the latter is clearly the effective bid which j makes to query i in run $\mathcal{R}$. Then, in run $\mathcal{R}$, no bid to i will surpass $\text{ebid}(i, j) = \text{bid}(i, j) \cdot (1 - p_j)$.

Section 4.2 presents an algorithm for General, using fake money; the above-stated assumption is used in its analysis, in particular in the proof of Claim 23. Section 4.4 shows that by upper bounding the fake money used in the worst case, we get an optimal algorithm for Small, again based on the above-stated assumption.

Algorithm 2. **($\mathcal{A}_3$: Algorithm for General)**

```
1. Initialization: $M \leftarrow \emptyset$, $W \leftarrow 0$ and $W_f \leftarrow 0$
   $\forall j \in A$, 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) $L_j \leftarrow B_j$.

2. Query arrival: When query $i$ arrives, do:
   (a) $\forall j \in A$ s.t. $(i,j) \in E$ and $L_j > 0$ do:
        i. $\text{ebid}(i,j) \leftarrow \text{bid}(i,j) \cdot (1 - p_j)$.
       ii. Offer effective bid of $\text{ebid}(i,j)$ 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: $u_i \leftarrow \text{bid}(i,j) \cdot (1 - p_j)$.
       ii. Update revenue: $r_j \leftarrow r_j + \text{bid}(i,j) \cdot p_j$.
      iii. Update matching: $M \leftarrow M \cup (i,j)$.
       iv. Update weight: $W \leftarrow W + \min\{L_j, \text{bid}(i,j)\}$ and
           $W_f \leftarrow W_f + \max\{0, \text{bid}(i,j) - L_j\}$.
        v. Update $L_j$: $L_j \leftarrow L_j - \min\{L_j, \text{bid}(i,j)\}$.
3. Output: Output matching $M$, real money spent $W$, and fake money spent $W_f$.
```

4.1 Structural Difficulties in General

To describe the structural difficulties in GENERAL, we provide three instances in Example 18. In order to obtain a completely unconditional result, we would need to adopt the following convention: assume bidder j has L_j money leftover and impression i just arrived. Assume that j's bid for i is $\text{bid}(i, j)$. If $\text{bid}(i, j) > L_j$, then j should not be allowed to bid for i, since j has insufficient money.

Under this convention, it is easy to see that even a randomized algorithm will accrue only $\$W$ expected revenue on at least one of the instances given in Example 18, provided it is greedy, i.e., if a match is possible, it does not rescind this possibility; the latter condition is a simple way of ensuring that the algorithm is "fine tuned" for a particular type of example. Note that the optimal for each instance is $\$2W$.

Example 18. Let $W \in \mathbb{Z}_+$ be a large number. We define three instances of GENERAL, each having two bidders, b_1 and b_2, with budgets of $\$W$ each. Instances I_1 and I_2 have $W + 1$ queries, where for the first W queries, both bidders bid \$1 each. For the last query, under I_1, b_1 bids $\$W$ and b_2 is not interested. Under I_2, b_2 bids $\$W$ and b_1 is not interested. Instance I_3 has $2W$ queries and both bidders bid \$1 for each of them.

Therefore, to obtain a non-trivial competitive ratio, bidder j must be allowed to bid for i even if $L_j < \text{bid}(i, j)$. This amounts to the use of free disposal, since j will be allowed to obtain query i for less money than its value for i. Next, let's consider a second convention: if $L_j < \text{bid}(i, j)$, then j will bid L_j for i. As stated in Remark 25, this convention is not supported by our proof technique, since Claim 23 fails to hold, breaking the proof of Lemma 22 and hence Lemma 24.

This led us to a third convention: if $L_j < \text{bid}(i, j)$, then j will bid L_j real money and $\text{bid}(i, j) - L_j$ "fake" money for i. As a result, the total revenue of the algorithm consists of real money as well as fake money; in Algorithm 2, these are denoted by W and W_f, respectively. The problem now is that Lemma 24, which compares the total revenue of the algorithm, namely $W + W_f$, with the optimal offline revenue, does not yield the competitive ratio of Algorithm 2. Remark 25 explains why our proof technique does not allow us to dispense with the use of fake money.

We note that when Algorithm 2 is run on instances of OBM, it reduces to RANKING. Therefore, it is indeed a (simple) extension of RANKING to GENERAL.

4.2 Algorithm for General

Algorithm 2, which will be denoted by $\mathcal{A}_3$, is an attempt at online algorithm for GENERAL. As stated in Section 4.1, because of the use of fake money, we will not be able to give a competitive ratio for it, instead, in Lemma 24, we will compare the sum of real and fake money spent by the algorithm with the real money spent by an optimal offline algorithm.

In algorithm $\mathcal{A}_3$, $L_j \in \mathbb{Z}_+$ will denote bidder j's *leftover budget*; it is initialized to B_j. At the arrival of query i, bidder j will bid for i if $(i, j) \in E$ and $L_j > 0$. In general, i will receive a number of bids. The exact procedure used by i to accept one of these bids is given in algorithm $\mathcal{A}_3$; its steps are self-explanatory. If i accepts j's bid then i is matched to j, the edge (i, j) is assigned a weight of $\text{bid}(i, j)$ and L_j is decremented by $\min\{L_j, \text{bid}(i, j)\}$.

Note that we do not require that there is sufficient left-over money, i.e., $L_j \geq \text{bid}(i, j)$, for j to bid for i. In case $L_j < \text{bid}(i, j)$ and i accepts j's bid, then $\text{bid}(i, j) - L_j$ of the money paid by j for i is *fake money*; this will be accounted for by incrementing W_f by $\text{bid}(i, j) - L_j$. The rest, namely L_j, is real money and is added to W. If $\text{bid}(i, j) \geq L_j$ and i accepts j's bid, then L_j becomes zero and j does not bid for any future queries. At the end of the algorithm, random variable W denotes the total real money spent and W_f denotes the total fake money spent.

The *offline optimal solution* to this problem is defined to be a matching of queries to advertisers that maximizes the weight of the matching; this is done with full knowledge of graph G. As stated in Remark 3, we will assume that under such a matching, P, the budget B_j of each bidder j is fully spent, i.e., $w(P) = \sum_{j=1}^{m} B_j$.

4.3 Analysis of Algorithm 2

Lemma 19.

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

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

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

□

Recall that for SINGLE-VALUED, we gave Lemma 6 and Corollary 7, which established a relationship between the available bidders for a query i in the two runs $\mathcal{R}$ and $\mathcal{R}_j$. These facts dealt with multisets and in Section 3, we have defined operations on multisets.

We will need Lemma 6 and Corollary 7 for analyzing Algorithm 2 as well, though the definitions of the multisets will be guided by the following: If bidder $k \in A$ has leftover money of L_k, as determined by Algorithm 2, then we will say that i has L_k copies of k *available* to it. Furthermore, if k's bid for i is $\text{bid}(i, k)$ and this bid is successful, then L_k will be decremented by $\min\{L_k, \text{bid}(i, k)\}$, as stated in Step 2(b)(v) of the algorithm, and the available copies of k for the next bidder will decrease accordingly.

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 copies of each bidder 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. Similarly, let $S(i)$ and $S_j(i)$ denote the multisets obtained by restricting $T(i)$ and $T_j(i)$ to the bidders that have edges to query i in graphs G and G_j, respectively.

We have assumed that Step (1) of Algorithm 2 has already been executed and a price p_k has been assigned to each good k. With probability 1, the prices are all distinct. Let F_1 be the multiset containing B_l copies of l for each $l \in A$ such that $p_l < p_j$. Similarly, let F_2 be the multiset containing B_l copies of l for each $l \in A$ such that and $p_l > p_j$.

Under the definitions and operations stated above, it is easy to check that Lemma 6 and Corollary 7 hold for Algorithm 2 as well. Therefore, Lemma 12 also carries over. Definition 8 needs to be modified to the following.

Definition 20. 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 $u_e = \min\{u_i,\ \text{bid}(i, j) \cdot \left(1 - \frac{1}{e}\right)\}$, where u_i is the utility of query i in run $\mathcal{R}_j$.

Definition 9 needs to be changed to the following.

Definition 21. Let $j \in A$. Let $i_1, \ldots, i_k$ be queries such that for $1 \leq l \leq k$, $(i_l, j) \in E$ and $\sum_{l=1}^{k} \text{bid}(i_l, j) = B_i$. Then $(j;\ i_1, \ldots, i_k)$ is called a B_j*-star*. Let X_j denote this B_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 B_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. The next lemma crucially uses the fact that p_j is independent of u_{e_l}; the reason for this fact is the same as in SINGLE-VALUED.

Lemma 22. *Let $j \in A$ and let $X_j = (j;\ i_1, \ldots, i_k)$ be a B_j-star. Then*

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

Proof. We will first lower bound $\mathbb{E}[r_j]$. Let $f_U(\text{bid}(i_1, j) \cdot z_1,\ \ldots\ , \text{bid}(i_k, j) \cdot z_k)$ be the joint probability density function of $(u_{e_1}, \ldots u_{e_k})$; clearly, $f_U(\text{bid}(i_1, j) \cdot z_1,\ \ldots\ , \text{bid}(i_k, j) \cdot z_k)$ 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, $\quad \mathbb{E}[r_j] =$

$$\int_{(z_1, \ldots, z_k)} \mathbb{E}[r_j \mid u_{e_1} = \text{bid}(i_1, j) \cdot z_1, \ldots, u_{e_k} = \text{bid}(i_k, j) \cdot z_k] \cdot f_U(\text{bid}(i_1, j) \cdot z_1, \ldots \text{bid}(i_k, j) \cdot z_k)\, dz_1 \ldots dz_k,$$

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

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

Furthermore, define

$$V(x) = \sum_{l=1}^{k} I_l(x) \cdot \text{bid}(i_l, j).$$

Claim 23. *Conditioned on $(u_{e_1} = \mathrm{bid}(i_1, j) \cdot z_1, \ldots, u_{e_k} = \mathrm{bid}(i_k, j) \cdot z_k)$, if $p_j = e^{x-1}$, where $x \in [0, 1]$, then the contribution to r_j in this run of algorithm $\mathcal{A}_3$ was $\geq p_j \cdot V(x)$.*

Proof. Suppose $I_l(x) = 1$, then $x < w_l$. In run $\mathcal{R}_j$, the maximum effective bid that i_l received has value $\mathrm{bid}(i_l, j) \cdot z_l$. In run $\mathcal{R}$, if on the arrival of query i_l, $L_j = 0$, i.e., j is already fully matched, then the contribution to r_j in this run was $B_j \cdot p_j$ and the claim is obviously true. If $L_j > 0$, then since $x < w_l$, $1 - p_j > z_l$. Therefore, by Corollary 7, query i_l will receive its largest effective bid from j. Hence, i_l will get matched to j and r_j will be incremented by $\mathrm{bid}(i_l, j) \cdot p_j$. The claim follows. □

By Claim 23,

$$\mathbb{E}[r_j \mid u_{e_1} = \mathrm{bid}(i_1, j) \cdot z_1, \ldots, u_{e_k} = \mathrm{bid}(i_k, j) \cdot z_k] \;\geq\; \int_0^1 V(x) \cdot e^{x-1}\, dx$$

$$= \sum_{l=1}^{k} \mathrm{bid}(i_l, j) \cdot \int_0^1 I_l(x) \cdot e^{x-1}\, dx \;=\; \sum_{l=1}^{k} \mathrm{bid}(i_l, j) \cdot \int_0^{w_l} e^{x-1}\, dx$$

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

Therefore, $\mathbb{E}[r_j] \;=$

$$\int_{(z_1, \ldots, z_k)} \mathbb{E}[r_j \mid u_{e_1} = \mathrm{bid}(i_1, j) \cdot z_1, \ldots, u_{e_k} = \mathrm{bid}(i_k, j) \cdot z_k] \cdot f_U(\mathrm{bid}(i_1, j) \cdot z_1, \ldots \mathrm{bid}(i_k, j) \cdot z_k)\, dz_1 \ldots dz_k,$$

$$\geq B_j \cdot \int_{(z_1, \ldots, z_k)} \sum_{l=1}^{k} \left(1 - \frac{1}{e} - z_l \right) \cdot f_U(\mathrm{bid}(i_1, j) \cdot z_1, \ldots \mathrm{bid}(i_k, j) \cdot z_k)\, dz_1 \ldots dz_k$$

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

By Lemma 12, $\mathbb{E}[u_{i_l}] \geq \mathbb{E}[u_{e_l}]$, for $1 \leq l \leq k$. Hence we get

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

□

Lemma 24. *Algorithm $\mathcal{A}_3$ satisfies*

$$\mathbb{E}\left[W + W_f \right] \;\geq\; \left(1 - \frac{1}{e} \right) \cdot w(P).$$

Furthermore, it is budget-oblivious.

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

$$w(P) = \sum_{j=1}^{m} B_j.$$

Let T_j denote the j-star, under P, corresponding to each $j \in A$. The expected weight of matching produced by $\mathcal{A}_3$ is

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

where the first equality uses Lemma 15, the second follows from linearity of expectation and the inequality follows by using Lemma 22.

Finally, Algorithm $\mathcal{A}_3$ is budget-oblivious because it does not need to know the budgets B_j for bidders j; it only needs to know during a run whether B_j has been exhausted. The lemma follows. $\square$

Remark 25. Let us consider the following two avenues for dispensing with the use of fake money altogether; we will show places where our proof technique breaks down for each one. Assume $L_j < \text{bid}(i, j)$.

1. Why not modify Step 2 of Algorithm 2 so that j's bid for i is taken to be L_j instead of $\text{bid}(i, j)$?
2. Why not modify Step 2(b)(i) so it sets u_i to $L_j \cdot (1 - p_j)$ rather than $B_j \cdot (1 - p_j)$

Under the first avenue, we cannot ensure $u_i \geq u_e$, since it may happen that $u_e > L_j \cdot (1 - p_j) = u_i$. The condition $u_i \geq u_e$ is used for deriving $\mathbb{E}[u_i] \geq \mathbb{E}[u_e]$, which is essential in the proof of Lemma 22.

To make the second avenue work, the proof of Claim 23 would need to be changed as follows: the last case, $L_j > 0$, will need to be split into the two cases given above. However, under Case 2, which applies if $L_j < \text{bid}(i, j)$, even though $p_j < p$, the largest effective bid that query i_l receives may not be the one from j, since the effective bid of j has value $L_j \cdot (1 - p_j) < \text{bid}(i_l, j) \cdot (1 - p_j)$. Therefore, i_l may not get matched to j, thereby invalidating Claim 23.

4.4 Algorithm for Small

We will use Lemma 24 to show that Algorithm 2 yields algorithms for SMALL by upper bounding the fake money used in the worst case. Their budget-obliviousness follows from that of Algorithm 2.

Conditional Theorem 26. Algorithm $\mathcal{A}_3$ is an optimal online algorithm for SMALL; furthermore, it is budget-oblivious.

Proof. Let I be an instance of SMALL.

$$W_f \leq \sum_{j \in A} \max_{(i,j) \in E} \{\text{bid}(i,j) - 1\}$$

Therefore,

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

where $\mu(I)$ is defined in Section 2. Now, by definition of SMALL,

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

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

Therefore

$$\lim_{n(I)\to\infty} \frac{W_f}{w(P)} = 0.$$

The theorem follows from Lemma 24. □

4.5 Experimental Results

The purpose of the experimental results is two-fold: first, to determine how often is the No-Surpassing Property violated on a typical instance and second, to evaluate the performance of Algorithm 2. The results are summarized in the four Tables given. The instances were generated using the following parameters:

- The number of advertisers and queries was 20 and 2,000, respectively.
- Budgets were picked in the range $[100, 2000]$.
- Bids were picked in the range $[1, 20]$. In addition, for each advertiser, we ensured that the bids were at most 0.02 times the budget, to ensure that bids were small compared to budgets.

The instances of SMALL were generated as follows. We picked the edge densities of the underlying graph to be 0.05, 0.1, 0.15, 0.2, 0.25, 0.5 and 0.8. For each edge density, 20 underlying graphs were constructed at random. Then, budgets were assigned randomly to each advertiser, in the range specified. Finally, for each advertiser, bids were assigned to incident edges randomly, in the range specified. The order of arrival of queries was picked at random. This resulted in one instance of SMALL.

For generating instances of SINGLE-VALUED, the underlying graphs were constructed as above. An instance of SINGLE-VALUED was obtained from such a graph as follows. For each advertiser j, the bid value b_j was picked randomly from the integral interval $[1, 20]$. Then a random number n_j was picked randomly from the interval $[100, 2000]$ and k_j was set to $\lfloor \frac{n_j}{b_j} \rfloor$.

For each instance of Small, Algorithm 2 was run 40 times, each time randomly setting p_j for each advertiser j. The average revenue of these runs was computed and was divided by the revenue of the MSVV Algorithm run on the same instance. Finally, for each edge density, the average of these ratios over all 20 instances was computed and noted in Table 1. A similar procedure was followed for instances of Single-Valued using Algorithm 1 and the MSVV Algorithm. These results are reported in Table 2.

Tables 3 compares, for all edge densities and instances of Small constructed above, Algorithm 2 to the greedy algorithm, under which each query is matched to the highest bid. Finally, Tables 4 compares the MSVV Algorithm to the greedy algorithm. Observe that the performances of Algorithm 2 and the MSVV Algorithm are far superior to that of the greedy algorithm, thereby indicating that these instances are difficult enough to not succumb to a trivial algorithm and lending more credence to the results reported in Tables 1 and 2.

Violations of the No-Surpassing Property were computed as follows. For each instance of Small and for each setting of random prices p_j for advertisers j, we determined if a violation occurred for each edge (i, j) of the underlying graph, where i is a query and j is an advertiser, as follows: check if the effective bid made to query i in run $\mathcal{R}_j$ is less than $\text{bid}(i, j) \cdot (1 - p_j)$; the latter being the effective bid which j makes to query i in run $\mathcal{R}$. If so, then if in run $\mathcal{R}$, the effective bid to i surpasses $\text{ebid}(i_l, j) = \text{bid}(i, j) \cdot (1 - p_j)$, then we say that a violation of the No-Surpassing property has occurred for edge (i, j). In Table 1, we have reported the percentage of edges for which violations happen.

As shown in Lemma 11, the No-Surpassing Property holds for Single-Valued. As a stress test of our code, we repeated the above experiment on instances of Single-Valued as well. The results are reported in Table 2; all entries are zero, as expected.

5 Discussion

The open question mentioned in the Introduction, of removing the assumption of no-surpassing property from our proof of Algorithm 2 for Small, deserves special attention because of its potential impact in the ad auctions marketplace. Another question is to place an upper bound on the expected fake money used, $\mathbb{E}\left[W_f\right]$, in Algorithm 2 and strengthen Lemma 24 to obtain a good bound on the competitive ratio of this algorithm for General. This seems a promising avenue for improving the bound for General from 0.5016, given in [HZZ20].

6 Acknowledgements

I wish to thank Asaf Ferber, Alon Orlitsky, Will Ma, Thorben Tröbst and Rajan Udwani for valuable comments and discussions. A big thanks to Jacob Tyler Bigham and Joseph Jin-Him Wong, undergrads at UCI, for conducting the extensive experiments reported in this paper.

References

[ABM19] Gagan Aggarwal, Ashwinkumar Badanidiyuru, and Aranyak Mehta. Autobidding with constraints. In *International Conference on Web and Internet Economics*, pages 17–30. Springer, 2019.

[AGKM11] 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.

[AS21] 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.

[BJN07] 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.

[BM08] Benjamin Birnbaum and Claire Mathieu. On-line bipartite matching made simple. *ACM Sigact News*, 39(1):80–87, 2008.

[DJK13] 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.

[DM22] 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.

[EFFS21] 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.

[GM08] Gagan Goel and Aranyak Mehta. Online budgeted matching in random input models with applications to adwords. In *SODA*, volume 8, pages 982–991, 2008.

[GS62] David Gale and Lloyd S Shapley. College admissions and the stability of marriage. *The American Mathematical Monthly*, 69(1):9–15, 1962.

[HT22] 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.

[HZZ20] 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.

[Ins19] Simons Institute. Online and matching-based market design, 2019. https://simons.berkeley.edu/programs/market2019.

[JMM+03] 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.

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

[KVV90] 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.

[Meh13] Aranyak Mehta. *Online matching and ad allocation*, volume 8. Now Publishers, Inc., 2013.

[MSVV07] Aranyak Mehta, Amin Saberi, Umesh Vazirani, and Vijay Vazirani. Adwords and generalized online matching. *Journal of the ACM (JACM)*, 54(5), 2007.

[Udw21] Rajan Udwani. Adwords with unknown budgets. *arXiv preprint arXiv:2110.00504*, 2021.

[Vaz21] Vijay V Vazirani. Online bipartite matching and adwords. *arXiv preprint arXiv:2107.10777*, 2021.

[Vaz22] Vijay V Vazirani. Online bipartite matching and adwords. In *47th International Symposium on Mathematical Foundations of Computer Science*, 2022.

Table 1: Ratio of Algorithm 2 to MSVV for SMALL

p_edge	Algorithm to MSVV ratio ± SD (%)	NSP violation mean ± SD (%)	NSP violation range
0.05	96.52 ± 11.70	0.01 ± 0.01	0.00, 0.03
0.1	98.45 ± 15.50	0.48 ± 0.04	0.42, 0.58
0.15	100.99 ± 16.66	0.76 ± 0.05	0.67, 0.90
0.2	101.32 ± 16.46	0.94 ± 0.04	0.86, 1.03
0.25	101.38 ± 16.82	1.06 ± 0.04	1.00, 1.13
0.5	100.89 ± 13.45	1.39 ± 0.06	1.29, 1.51
0.8	101.17 ± 15.55	1.61 ± 0.03	1.56, 1.67

Table 2: Ratio of Algorithm 1 to MSVV for SINGLE-VALUED

p_edge	Algorithm to MSVV ratio ± SD (%)	NSP violation mean ± SD (%)	NSP violation range
0.05	99.99 ± 2.95	0.00 ± 0.00	0.00, 0.00
0.1	99.93 ± 0.67	0.00 ± 0.00	0.00, 0.00
0.15	99.80 ± 0.60	0.00 ± 0.00	0.00, 0.00
0.2	100.54 ± 1.15	0.00 ± 0.00	0.00, 0.00
0.25	99.85 ± 0.78	0.00 ± 0.00	0.00, 0.00
0.5	100.04 ± 1.15	0.00 ± 0.00	0.00, 0.00
0.8	100.44 ± 1.79	0.00 ± 0.00	0.00, 0.00

Table 3: Ratio of Algorithm 2 to Greedy for SMALL

p_edge	Ratio of Algorithm to Greedy mean ± SD (%)
0.05	95.84 ± 0.53
0.1	101.32 ± 0.73
0.15	106.51 ± 1.08
0.2	110.29 ± 1.69
0.25	113.85 ± 1.81
0.5	131.78 ± 4.38
0.8	152.75 ± 10.59

Table 4: Ratio of MSVV to Greedy for SMALL

p_edge	Ratio of MSVV to Greedy ± SD (%)
0.05	100.12 ± 0.11
0.1	105.02 ± 1.32
0.15	107.46 ± 1.63
0.2	109.84 ± 1.77
0.25	114.12 ± 2.09
0.5	133.92 ± 5.71
0.8	152.63 ± 9.05