

Primal-Dual Schemes for Online Matching in Bounded Degree Graphs

Ilan Reuven Cohen  

Bar-Ilan University, Ramat Gan, Israel

Binghui Peng 

Columbia University, New York, NY, USA

Abstract

We explore various generalizations of the online matching problem in a bipartite graph G as the b -matching problem [8], the allocation problem [5], and the AdWords problem [13] in a beyond-worst-case setting. Specifically, we assume that G is a (k, d) -bounded degree graph, introduced by Naor and Wajc [14]. Such graphs model natural properties on the degrees of advertisers and queries in the allocation and AdWords problems. While previous work only considers the scenario where $k \geq d$, we consider the interesting intermediate regime of $k \leq d$ and prove a tight competitive ratio as a function of k, d (under the small-bid assumption) of $\tau(k, d) = 1 - (1 - k/d) \cdot (1 - 1/d)^{d-k}$ for the b -matching and allocation problems. We exploit primal-dual schemes [6, 3] to design and analyze the corresponding tight upper and lower bounds. Finally, we show a separation between the allocation and AdWords problems. We demonstrate that $\tau(k, d)$ competitiveness is impossible for the AdWords problem even in (k, d) -bounded degree graphs.

2012 ACM Subject Classification Theory of computation → Online algorithms

Keywords and phrases Online Matching, Primal-dual analysis, bounded-degree graph, the AdWords problem

Digital Object Identifier 10.4230/LIPIcs.ESA.2023.35

Funding *Ilan Reuven Cohen:* Supported by the Israel Science Foundation grant No. 1737/21.

Binghui Peng: Supported by NSF IIS-1838154, CCF-2106429, CCF-2107187, CCF-1763970, CCF-2212233.

1 Introduction

Ad auctions have emerged as a powerful tool for online advertisers seeking to reach targeted audiences through search engines and other digital platforms. Therefore, there has been much interest in optimizing their revenue generation in recent years. In an ad auction, advertisers bid on specific keywords or phrases relevant to their products or services. When users query search engines, in addition to the algorithmic search results, it also displays sponsored ads that match the ad auctions. These ads are rapidly sold or assigned to potential buyers (advertisers). In their seminal work, Mehta et al. [13] developed a model for optimizing the allocation of ad auctions, building upon the concept of online bipartite matching of Karp et al. [10]. There are n bidders, where each bidder $i \in \{1, \dots, n\}$, has a known daily budget $B(i)$. Ad auctions arrive online, one at a time, and each bidder i provides a bid $b(i, j)$ for buying the product j (displaying the ad). Once the bids are received, the seller's algorithm must allocate the ad to one of the interested bidders, and this decision is irrevocable. The seller's objective is to maximize the total revenue accumulated from the ad auctions. To accomplish this, Mehta et al. [13] proposed a (tight) deterministic algorithm, which is $(1 - 1/e)$ -competitive in cases where the budget of each bidder is relatively larger than the bids. This is a realistic assumption in the ad auction scenario.

Consequently, a loss of $1/e$ can equate to billions of dollars in potential revenue loss each year. In addition, the $1 - 1/e$ hardness result of Karp et al. [10] for online matching is prevalent in all the problem extensions. Therefore, researchers have been inspired to pursue a more comprehensive theoretical understanding that extends beyond the worst-case scenarios.

One line of research considers stochastic arrival models, which include random order arrival and iid arrival models. In the random order model, a fixed input graph is prepared in advance, and the online vertices are randomly arranged. On the other hand, the iid arrival model involves drawing online vertices from a known or unknown distribution. With these arrival models, researchers were able to demonstrate results that outperform the $1 - 1/e$ competitive ratio in both online matching and vertex-weighted matching, as demonstrated by various sources [4, 7, 9, 11, 12].

In this work, we consider a different line of research, which assumes structural properties commonly encountered in ad allocation instances used for targeted advertising. Such properties were first identified and formalized by Naor and Wajc [14]. We assume that G is a bipartite (k, d) -bounded-degree graph (for brevity's sake, we use the term (k, d) -graphs).

Specifically, we assume that advertisers are interested in a large number of ad slots (i.e. $\sum_j b(i, j) \geq k \cdot B(i)$ for all i) and every ad slot is of interest to a relatively small number of advertisers (i.e. $|N(j)| = |\{i : b(i, j) > 0\}| \leq d$ for all j).

This structural assumption is relatively natural for ad allocation graphs. First, ad campaigns often target relatively small, specific population segments, resulting in users belonging to fairly few segments. When coupled with the limited number of active campaigns, this creates a limited pool of ads that may be shown to a particular user, justifying the assumption of a small degree for ad slots. Second, advertisers generally aim to target large population segments. However, they often allocate insufficient budget to display ads to all users within a segment. This leads to the high-degree assumption on the offline side since every page view by a specific, targeted user corresponds to a vertex in the graph.

Related work. Buchbinder et al. [5] developed a primal-dual algorithm for AdWords that attains a competitive ratio of $(1 - 1/c)(1 - R_{\max})^{1/R_{\max}}$, where $c = (1 + R_{\max})^{1/R_{\max}}$. As $R_{\max} \rightarrow 0$, the competitiveness tends to $1 - 1/e$. Buchbinder et al. also examined a setting where the degree of each incoming query was upper bounded by d (i.e., $(1, d)$ -graph) and produced an algorithm with a competitive ratio of nearly $1 - (1 - 1/d)^d$ for the AdWords with equal bids per advertiser (a.k.a, the allocation problem). Azar et al. [3] showed that this ratio is the best possible, also for randomized algorithms. The expression $1 - (1 - 1/d)^d$ is always greater than $1 - 1/e$ but approaches the latter as d increases. Naor and Wajc [14] introduced the (k, d) -graphs class and applied it to the AdWords problem and online bipartite matching. For $k \geq d$, they developed deterministic online algorithms achieving a competitive ratio of $1 - (1 - 1/d)^k$ for online bipartite matching and its vertex-weighted extension, where all edges connected to an offline vertex i have the same weight. They also showed that this ratio holds for AdWords with equal bids per advertiser, where each advertiser has the same bid for all relevant keywords. The ratio of $1 - (1 - 1/d)^k$ is the best possible for online bipartite matching and the vertex-weighted extension if $k \geq d$. For AdWords with arbitrary bids, Naor and Wajc provided an algorithm with a competitive ratio of $(1 - R_{\max})(1 - (1 - 1/d)^k)$. They also demonstrated that if $k \geq d$, the best possible competitiveness is upper-bounded by $(1 - R_{\max})(1 - (1 - 1/d)^k/R_{\max})$. As k/d increases, the expression $1 - (1 - 1/d)^k$ tends to 1, and for $k \simeq d$ it approaches $1 - 1/e$.

A recent work [2], studied the online b -matching problem in (k, d) -graphs for $k \geq d$. In this scenario, each vertex a located on the left-hand side of the bipartite graph represents a server with a capacity of b_a , which implies that it can be matched with up to b_a incoming

	$\alpha = 1/3$	$\alpha = 1/2$	$\alpha = 2/3$	$\alpha = 3/4$
$d = 12$	0.66765	0.70335	0.76464	0.80744
$d = 24$	0.66258	0.69997	0.76286	0.80634
$d = 48$	0.69997	0.69833	0.76200	0.80581
$d = 120$	0.65868	0.69737	0.76149	0.8055
$d \rightarrow \infty$	0.65772	0.69673	0.76116	0.8053

■ **Figure 1** Comparison of $\tau(k, d)$ for various $d, k = \alpha \cdot d$ values.

requests that appear as right-hand side vertices. The objective is to maximize the size of the generated matching. Their results also hold for the vertex-weighted problem extension and, thus, for AdWords and auction problems in which each bidder issues individual, equally valued bids. They designed deterministic online algorithms for optimal competitiveness, for $k \geq d$ instances. Later, in [1], they showed a non-trivial extension for the AdWords problem, i.e., bids of arbitrary value. Specifically, they showed an algorithm for the general AdWords problem that achieves competitiveness arbitrarily close to 1 for $k \geq d$ using the small-bids assumption.

Our Contributions. We examine the interesting scenarios of (k, d) -graphs where $k \leq d$. First, we study the b -matching problem for $k \in \mathbb{N}$. Our algorithm is based on a fractional algorithm for b -matching. In Section 3, we provide a fractional algorithm with a competitive ratio of $\tau(k, d)$, where $\tau(k, d) = 1 - \frac{d-k}{d} \left(1 - \frac{1}{d}\right)^{d-k}$. The algorithm is based on two steps; The first ensures that each offline vertex's fractional load at the end of the algorithm is at least k/d , while the second step uses the water-level algorithm to allocate the remaining volume. We analyze the algorithm using the primal-dual scheme and a carefully chosen potential function, $f^{(k,d)}$. Second, we prove that the bound $\tau(k, d)$ is tight. We use the primal-dual techniques from [3] to analyze a class of (k, d) -graphs where any online algorithm is at most $\tau(k, d)$ -competitive.

Next, for arbitrary $k \in \mathbb{R}_+$, let $\alpha = k/d \leq 1$, we prove that for AdWords in the equally-valued bids case, it is possible to achieve almost $\tau(\alpha)$ -competitive values (under the small bid assumption), where $\tau(\alpha) = 1 - (1 - \alpha) \cdot e^{\alpha-1}$. It is evident that $\lim_{d \rightarrow \infty, k = \alpha \cdot d} \tau(k, d) = \tau(\alpha)$. Figure 1 presents a comparison of the bound of $\tau(k, d)$ as a function of $\alpha = k/d$ and d . The algorithm uses a similar approach to our fractional matching algorithm on (k, d) -graph and uses a 'smoother' potential function $f^{(\alpha)}$ as a function only of α . We round the fractional algorithm outputs using the algorithm of [5].

Finally, we prove that achieving a $\tau(\alpha)$ -competitive algorithm for the AdWords problem for bids of arbitrary value is impossible. To do that, we significantly extend the lower bounds of [3], which essentially uses a single scenario, to a multi-scenario instance; we show that the primal-dual techniques can be adapted to analyze the multi-scenario. Subsequently, we prove an improved upper bound by carefully choosing such a multi-scenario instance.

2 Preliminaries

The AdWords problem

The online ad auctions problem comprises a group of $L = \{1, \dots, n\} = [n]$ buyers, where bidder i has a daily budget of $B(i)$. In the online setting, a group of $R = [m]$ products arrives one by one, and each buyer i places a bid $b(i, j)$ for each product j . The objective is to allocate the products to buyers to maximize the seller's revenue. The allocation can be

Primal:	Dual:
$\max \sum_{j \in R} \sum_{i \in N(j)} b(i, j) \cdot x_{i,j}$ $\sum_{j \in N(i)} b(i, j) \cdot x_{i,j} \leq B(i), \quad \forall i \in L \quad (y_i)$ $\sum_{i \in N(j)} x_{i,j} \leq 1, \quad \forall j \in R \quad (z_j)$ $x_{i,j} \geq 0, \quad \forall i, j$	$\min \sum_{i \in L} B(i) \cdot y_i + \sum_{j \in R} z_j$ $b(i, j) \cdot y_i + z_j \geq b(i, j), \quad \forall i, j \quad (x_{i,j})$ $y_i \geq 0, \quad \forall i \in L$ $z_j \geq 0, \quad \forall j \in R$

Figure 2 The fractional AdWords problem (the primal) and the corresponding dual problem.

integral, where a product goes to a single buyer, or fractional, where products can be divided among several buyers. However, the sum of the fractions cannot exceed 1 for each product. The revenue received from a buyer i is the minimum between the sum of the costs of the products allocated to the buyer (times the allocated fraction) and the buyer's total budget. The problem can be formulated as a linear programming problem where the variable $x_{i,j}$ denote the fraction of product j allocated to buyer i . The objective is to maximize the total seller revenue, and the constraints guarantee that the sum of fractions for each product does not exceed 1 and each buyer's budget is not exceeded. See Figure 2 for the corresponding fractional Linear Program and its dual.

► **Definition 1** ((k, d)-bounded graph [14]). A bipartite graph $G = (L, R, E)$ is (k, d) -bounded if each left vertex $i \in L$ has a degree $d(i) \geq k$, and every right vertex $j \in R$ has a degree $d(j) \leq d$. For ad allocations, we replace $d(i) \geq k$ with the property $\sum_j b(i, j) \geq k \cdot B(i)$.

Rounding the fractional solution. For the AdWords problem with equal bids per advertiser, in [5], they presented an algorithm that rounds the online fractional solution, such that if each product's bid price is small compared to the total bidder budget (the small bid assumption), then this rounding phase only reduces the revenue by a factor of $1 - o(1)$ compared to the fractional solution revenue. Formally, let $b_{\max} = \max_j b(j)$ denote the maximum bid price, and $B_{\min} = \min_i B(i)$.

► **Theorem** (Theorem 5.4 in [5]). There exists a deterministic online rounding algorithm, where the integral allocation algorithm's revenue is at least $1 - o(1)$ times the revenue of the fractional solution, provided that: $(1 + b_{\max}) \sqrt{\frac{\ln 2n}{B_{\min}}} = o(1)$

3 Matching on bounded-degree graphs

In this section, we prove the tight bounds for online fractional bipartite matching on bounded-degree graphs.

The b-matching problem

Consider a bipartite graph $G = (L \cup R, E)$, where L represents servers and R represents requests. The set of servers, L , is known beforehand, and each server $i \in L$ has an individual capacity $B(i)$, indicating that it can be matched with up to $B(i)$ requests. Requests arrive online, one by one, and when a new request $j \in R$ arrives, its incident edges are revealed, and

it must be immediately and irreversibly matched to an eligible server, provided one exists. The objective is to maximize the number of matching edges. According to the AdWords formulation, we have $b(i, j) = 1$ if $(i, j) \in E$ and 0 otherwise.

A fractional version of the bipartite matching problems allows for every node $j \in R$ to be allocated partially with nodes $i \in N(j)$ in fractions $x_{i,j}$. We study the case where G is a (k, d) -graph. Let $\tau(k, d) = 1 - \frac{d-k}{d} (1 - \frac{1}{d})^{d-k}$.

► **Theorem 2.** *For the fractional online b -matching problem in (k, d) -graphs, where $k \leq d$ and $k, d \in \mathbb{N}_+$, there exists a $\tau(k, d)$ -competitive algorithm.*

In Section 4, we prove the result is tight for any (k, d) graph. Note that we may apply the rounding algorithm of [5] (Theorem 5.4) to the b -matching problem.

3.1 The Two-Step-Water-Level algorithm

For a vertex $i \in L$, denote the level of vertex i as $\ell(i) = \frac{\sum_{j \in N(i)} x_{ij}}{B(i)}$. Note that, for (k, d) -graphs, a naive online algorithm can ensure that the level of any $i \in L$ at the end of the algorithm is k/d . This is achieved by assigning each online vertex a $1/d$ fraction to each neighbor. Since the degree of any $j \in R$ is at most d , the algorithm is feasible. In addition, since for any vertex $i \in L$, the degree is at least $k \cdot B(i)$, its level at the end of the algorithm will be k/d . Accordingly, our algorithm consists of two steps. For the allocation of an online vertex $j \in R$, the first step fractionally allocates the vertex's volume to vertices in $i \in N(j)$ such that their degree is less or equal to $k \cdot B(i)$. As a result of this step, a level of at least k/d is guaranteed at the end of the algorithm's run for all $i \in L$. The second step runs the water-level algorithm, continuously raising the level on the current set of minimum-level vertices in $N(j)$ on the remainder of the vertex j 's volume. We use the notations $d^j(i)$ and $\ell^j(i)$ for the degree and level of vertex i after vertex j 's arrival, respectively. Given a function $f^{(k,d)} : [0, 1] \rightarrow [0, 1]$, which we will describe later, the algorithm also maintains a solution to the dual variables.

■ **Algorithm 1** The Two-Step-Water-Level algorithm.

Upon the arrival of a new vertex $j \in R$:

1. For each vertex $i \in N(j)$, such that $d^j(i) \leq k \cdot B(i)$
Increase (if necessary) the level of i to $d^j(i)/d$
 2. Using the leftover volume:
Increase the level of vertices in $N(j)$ using the water-level algorithm
 3. Update the dual variables, let $\ell(j) = \min_{i \in N(j)} \ell^j(i)$ and $\hat{\ell}(j) = \max(k/d, \ell(j))$.
Set $y_i = f^{(k,d)}(\ell(i))$, for $i \in N(j)$
Set $z_j = 1 - f^{(k,d)}(\hat{\ell}(j))$
-

3.2 The algorithm's feasibility

We will prove that for a proper $f^{(k,d)}$, (for brevity's sake, we use f for $f^{(k,d)}$ for the rest of the section) the primal and dual solutions of the **Two-Step-Water-Level** (TSWL) algorithm are feasible.

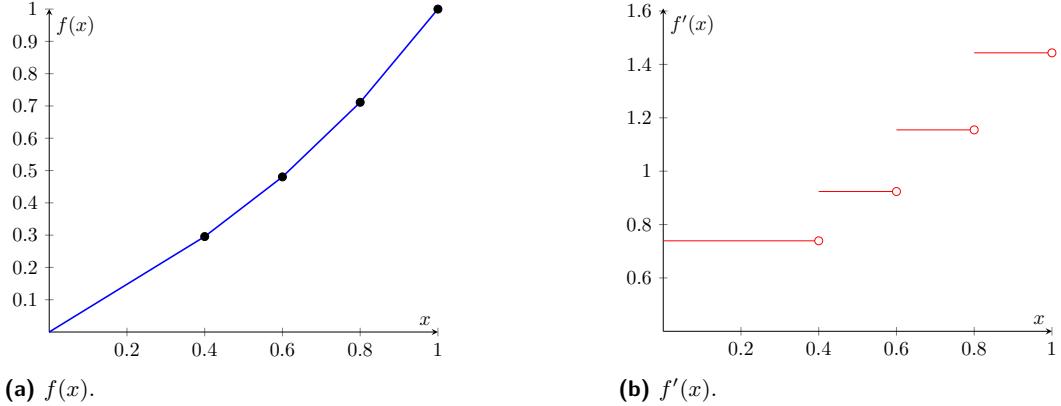


Figure 3 The function $f^{(k,d)}(x)$ for $k = 2$ and $d = 5$.

► **Lemma 3.** *Given a function $f : [0, 1] \rightarrow [0, 1]$, where f is a monotone non-decreasing function, the primal and dual solutions of the TSWL Algorithm are feasible at the end of the algorithm.*

Proof. First, we will show that the algorithm does not over-allocate online vertex j in step 1. We assume that the algorithm is feasible until the arrival of vertex j . For any vertex $i \in N(j)$, its degree increases by 1, and by the algorithm definition, its level prior to the assignment of vertex i is at least $\frac{d^j(i)-1}{d}$. Therefore, for any such i , x_{ij} after the first step is at most $\frac{d^j(i)}{d} - \frac{d^j(i)-1}{d} = \frac{1}{d}$, and by the definition of (k, d) -graphs, $|N(j)| \leq d$. Therefore, vertex j fractional allocation at step 1 is at most $|N(j)| \cdot \frac{1}{d} \leq 1$. By the definition of step 1, and since G is (k, d) -graph, at the end of the sequence, the level of any vertex $i \in L$ is at least k/d .

To prove that the dual is feasible, we need to verify for $i \in N(j)$ that $y_i + z_j \geq 1$ ($b(i, j) = 1$). If $\ell(j) \geq k/d$, then

$$y_i + z_j = f(\ell(i)) + 1 - f(\ell(j)) \geq f(\ell^j(i)) + 1 - f(\ell(j)) \geq 1,$$

where the first inequality is since $\ell(i) \geq \ell^j(i)$ and f is a monotone, non-decreasing function, and the second inequality is because, by definition, $\ell^j(i) \geq \ell(j)$. If $\ell(j) < k/d$, by definition $\hat{\ell}(j) = k/d$, then we have

$$y_i + z_j = f(\ell(i)) + 1 - f(\hat{\ell}(j)) \geq f(k/d) + 1 - f(k/d) \geq 1,$$

the first inequality is since the level of any vertex $i \in L$ at least k/d and the end of the algorithm, and since f is a monotone, non-decreasing function. ◀

3.3 The potential function

For a function f , we denote $f'_+(x)(f'_(x))$ the right (left) derivative of f at point x .

► **Lemma 4.** *For any $k \leq d, k \in \mathbb{N}_+$ there exists $f^{(k,d)} = f : [0, 1] \rightarrow [0, 1]$, such that*

- f, f' are monotone, non-decreasing functions.
- $f(0) = 0, f(1) = 1$.
- $1 - f\left(\frac{d-i}{d}\right) + f'_-\left(\frac{d-i}{d}\right) = \frac{1}{\tau(k,d)}$, for $i \in \{0, \dots, d-k\}$.

Proof. Let $\beta = 1/\tau(k, d)$. We define $f[0, 1] \rightarrow [0, 1]$ as a piece-wise linear function.

$$f'_+ \left(\frac{d-i+1}{d} \right) = f'_- \left(\frac{d-i}{d} \right) = \beta \cdot \left(\frac{d-1}{d} \right)^i, \quad \text{for } i \in \{0, \dots, d-k-1\},$$

and

$$f'_+(0) = f'_- \left(\frac{k}{d} \right) = \beta \cdot \left(\frac{d-1}{d} \right)^{d-k}.$$

By definition, f' is a monotone, non-decreasing function, and since $f'(x) > 0$, f is a monotone, increasing function. We set $f(0) = 0$, we have

$$\begin{aligned} f(1) &= f(0) + \frac{k}{d} \cdot f'_- \left(\frac{k}{d} \right) + \frac{1}{d} \cdot \sum_{i=0}^{d-k-1} f'_- \left(\frac{d-i}{d} \right) \\ &= 0 + \frac{k}{d} \cdot \beta \cdot \left(\frac{d-1}{d} \right)^{d-k} + \frac{1}{d} \cdot \sum_{i=0}^{d-k-1} \beta \cdot \left(\frac{d-1}{d} \right)^i \\ &= \beta \cdot \left(\frac{k}{d} \cdot \left(\frac{d-1}{d} \right)^{d-k} + 1 - \left(\frac{d-1}{d} \right)^{d-k} \right) = 1 \end{aligned}$$

Finally, we show by induction that for $i \in \{0, \dots, d-k\}$, we have $1 - f \left(\frac{d-i}{d} \right) + f'_- \left(\frac{d-i}{d} \right) = \beta$.
For $i = 0$:

$$1 - f \left(\frac{d-i}{d} \right) + f'_- \left(\frac{d-i}{d} \right) = 1 - 1 + \beta \cdot \left(\frac{d-1}{d} \right)^0 = \beta$$

and for $i \in \{1, \dots, d-k\}$:

$$\begin{aligned} 1 - f \left(\frac{d-i-1}{d} \right) + f'_- \left(\frac{d-i-1}{d} \right) \\ &= 1 - f \left(\frac{d-i}{d} \right) + \frac{1}{d} \cdot f'_- \left(\frac{d-i}{d} \right) + \frac{d-1}{d} \cdot f'_- \left(\frac{d-i}{d} \right) \\ &= 1 - f \left(\frac{d-i}{d} \right) + f'_- \left(\frac{d-i}{d} \right) = \beta. \end{aligned}$$
◀

3.4 The algorithm's competitive ratio

We are now able to complete the proof of Theorem 2. By demonstrating that the value of the primal is at least $\tau(k, d) = \frac{1}{\beta}$ times the value of the dual solution, we conclude by the weak duality that the TSWL is $\tau(k, d)$ -competitive.

We prove it by bounding the ratio of the increment of the primal and the dual objectives for every arrival of a vertex $j \in R$. We denote $\Delta P(\Delta D)$ as the primal (dual) value increment. Moreover, $\Delta D = \Delta L + \Delta R$, where $\Delta L(\Delta R)$ refers to the increment from the left side (right side). Now, we have the following key lemma:

► **Lemma 5.** *For every arrival of a vertex $j \in R$, we have $\Delta D \leq \beta \cdot \Delta P$.*

Proof. Given a vertex $j \in R$, let $t = \ell(j) \cdot d$ and $t^* = \lceil t \rceil$. We divide it into three cases:

Case 1: $\Delta P < 1$. Here, $t = d$ and vertex j 's neighbors have been fully matched, i.e., $\ell(i) = 1$ for all $i \in N(j)$. Then we would set $\Delta R = z_u = 1 - f(1) = 0$, thus we and $\Delta L \leq f'_-(1) \cdot \Delta P$ since f' is monotone, non-decreasing, therefore:

$$\Delta D = \Delta L + \Delta R \leq f'_-(1) \cdot \Delta P + 0 = f'_-(1) \cdot \Delta P = \beta \cdot \Delta P$$

Case 2: $\Delta P = 1$ and $t > k$. In this case, we know that $z_j = 1 - f(t/d)$. Next, let V_1 be the increment volume for j 's neighbors between $t^* - 1$ and t^* . And Let $V_2 = 1 - V_1$, i.e., the total volume of the increment for j 's neighbors below $t^* - 1$. We have

$$\begin{aligned}\Delta L &\leq V_1 \cdot f'_-(\frac{t^*}{d}) + V_2 \cdot f'_-(\frac{t^*-1}{d}) \\ &= V_1 \cdot f'_-(\frac{t^*}{d}) + V_2 \cdot \frac{d-1}{d} \cdot f'_-(\frac{t^*}{d}) \\ &= \frac{d-1}{d} \cdot f'_-(\frac{t^*}{d}) + \frac{1}{d} \cdot V_1 \cdot f'_-(\frac{t^*}{d}) \\ &\leq \frac{d-1}{d} \cdot f'_-(\frac{t^*}{d}) + \frac{1}{d} \cdot (t - t^* + 1) \cdot f'_-(\frac{t^*}{d}) \\ &= f'_-(\frac{t^*}{d}) + \frac{1}{d} \cdot (t - t^*) \cdot f'_-(\frac{t^*}{d})\end{aligned}$$

where the first inequality is because f' is a monotone, non-decreasing function, and the last inequality holds since we know that (note that $j \in R$, therefore $d(j) \leq d$)

$$V_1 \leq \left(\frac{t}{d} - \frac{t^*-1}{d} \right) \cdot d(j) \leq \left(\frac{t}{d} - \frac{t^*-1}{d} \right) \cdot d = t - t^* + 1$$

And, by f 's definition:

$$\Delta R = 1 - f(\frac{t}{d}) = 1 - f(\frac{t^*}{d}) - (\frac{t}{d} - \frac{t^*}{d}) \cdot f'_-(\frac{t^*}{d}) = 1 - f(\frac{t^*}{d}) - \frac{1}{d} \cdot (t - t^*) \cdot f'_-(\frac{t^*}{d}).$$

Therefore,

$$\Delta D = \Delta R + \Delta L \leq 1 - f(\frac{t^*}{d}) + f'_-(\frac{t^*}{d}) = \beta.$$

Case 3: $\Delta P = 1$ and $t \leq k$. By the algorithm definition, $\Delta R = z_j = 1 - f(k/d)$, and $\Delta L = f'_-(k/d)$. Thus, we have

$$\Delta D = \Delta R + \Delta L = 1 - f(\frac{k}{d}) + f'_-(\frac{k}{d}) = \beta. \quad \blacktriangleleft$$

4 Upper Bounds for fractional matching

In this section, we provide an instance indicating that $\tau(k, d)$ is the best possible competitive ratio that can be achieved by any online algorithm.

► **Theorem 6.** *For fractional online matching in (k, d) -graphs, no online algorithm can be better than $\tau(k, d)$ -competitive. This upper bound also holds for randomized algorithms.*

Proof. The hard instance is inspired by [3]. More concretely, we construct a parameterized family of primal linear programs based on a candidate collection of input sequences for proving the lower bound, where the objective function corresponds to optimizing the competitive ratio. Given d, k , our instance is a bipartite graph $G(L, R, E)$, where the number of vertices is $n = |L| = |R|$. Given a deterministic online fractional algorithm, the adversary sequence has d phases in total. Wherein in the first k phases, the adversary divides L into n/d groups, each containing d vertices, and introduces a single online vertex adjacent to each group. This guarantees $d(i) \geq k$ for all $i \in L$ and that the offline can match k vertices from each group. Accordingly, the adversary sets $L_{k+1} \subset L$ by dropping each group's

k -lowest load vertices (according to the online algorithm assignment). Next, in phase $t \in \{k+1, \dots, d-1\} = [k+1, d-1]$, the adversary divides L_t into $|L_t|/d$ groups and introduces a single online vertex adjacent to each group. The adversary defines $L_{t+1} \subset L_t$ by dropping the lowest load vertex from each group. In the last phase, for each vertex $i \in L_d$, the adversary introduces an online adjacent vertex.

Formally, the adversary defines for each phase $t \in [d]$, $L_t \subseteq L$, where all the new edges in phase t would be connected only to L_t , and the adversarial would set $L_d \subseteq L_{d-1} \subseteq \dots \subseteq L_1 = L$. Moreover, the adversary groups each set L_t into $|L_t|/d$ disjoint groups, each of size d , denoted as $L_{t,s}$, for $s \in [|L_t|/d]$. In phase $t \in [d-1]$, $|L_t|/d$ online vertices arrive, and the s^{th} ($s \in [|L_t|/d]$) vertex has edges to the offline vertices in the group $L_{t,s}$. For $t \in [k]$, $L_t = L$, and the adversary uses the same (arbitrary) grouping, i.e., $L_{t_1,s} = L_{t_2,s}$ for $t_1, t_2 \in [k]$, $s \in [n/d]$. According to the online algorithm assignment, the adversary sets L_{k+1} . Specifically, for $s \in [n/d]$, let $L_{k,s}^e \subset L_{k,s}$ the subset of the k -lowest load vertices in $L_{k,s}$ after phase k and $L_{k,s}^m = L_{k,s} \setminus L_{k,s}^e$ (the subset of the $(d-k)$ -highest load vertices in $L_{k,s}$). We define $L_{k+1} = \cup_{s=1}^{n/d} L_{k,s}^m$. note that, $|L_{k+1}| = (1 - k/d) |L_k|$. Similarly, the adversary sets L_{t+1} , for $t \in [k+1, d-1]$ as follow: For $s \in \frac{|L_t|}{d}$, let $L_{k,s}^e \subset L_{k,j}$ be the subset containing the lowest total load vertex in $L_{t,s}$ after phase t and $L_{t,s}^m = L_{t,s} \setminus L_{t,s}^e$. We define $L_{t+1} = \cup_{s=1}^{|L_t|/d} L_{t,s}^m$. Note that, $|L_{t+1}| = (1 - 1/d) |L_t|$ for $t \in [k+1, d-1]$. Finally, in phase d , for each vertex $i \in L_d$, the adversary introduces an online adjacent vertex. We set n as a large constant, ensuring that $|L_t|$ for $t \in [d]$ is a multiple of d .

Clearly, the optimal matching size is n since it matches online vertices of phases $t \in [k]$ to the vertices $L_{k,s}^e$ (for $s \in [n/d]$), and for $t \in [k+1, d-1]$ match the s^{th} online vertex of phase t to the vertex in $L_{t,s}^e$. The rest of the L_d vertices are matched in the last phase.

4.1 The primal linear program

We would use $x_k^m(x_k^e)$ as the average load assigned in phases $[k]$ on vertices in $L_{k,s}^m(L_{k,s}^e)$ for $s \in [n/d]$. Similarly, for $t \in [k+1, d-1]$ we use $x_t^m(x_t^e)$ as the average load assigned in phase t to vertices in $L_{t,s}^m(L_{t,s}^e)$. For $t \in [k+1, d-1]$, we use $u_t^m(u_t^e)$ as the average total load assigned before phase t to vertices in $L_{t,s}^m(L_{t,s}^e)$, and u_d^m as the average total load assigned before phase d to L_d .

$$\begin{aligned}
 & \max \frac{|L_k|}{d}((d-k)x_k^m + k \cdot x_k^e) + \sum_{t=k+1}^{d-1} \frac{|L_t|}{d}(x_t^e + (d-1)x_t^m) + |L_d|(1 - u_d^m) \\
 & (d-k) \cdot x_k^m + k \cdot x_k^e \leq k & (v_k) \\
 & (d-1) \cdot x_t^m + x_t^e \leq 1 & \forall t \in [k+1, d-1], \quad (v_t) \\
 & x_k^e \leq x_k^m & (m_k) \\
 & x_t^e + u_t^e \leq x_t^m + u_t^m & \forall t \in [k+1, d-1], \quad (m_t) \\
 & d \cdot x_k^m \leq (d-1) \cdot u_{k+1}^m + u_{k+1}^e & (c_k) \\
 & d \cdot (x_t^m + u_t^m) \leq (d-1) \cdot u_{t+1}^m + u_{t+1}^e & \forall t \in [k+1, d-2], \quad (c_t) \\
 & d \cdot (x_{d-1}^m + u_{d-1}^m) \leq d \cdot u_d^m & (c_{d-1}) \\
 & x_t^m, u_t^m, x_t^e, u_t^e \geq 0 & \forall t \in [k, d]
 \end{aligned}$$

Constraint v_k implies that the total volume in vertices in L_k^m (note, $\frac{|L_k^m|}{|L_k|} = \frac{d-k}{d}$) plus the total volume of vertices in L_k^e (note, $\frac{|L_k^e|}{|L_k|} = \frac{k}{d}$) does not exceed the total volume in the first k phases. Constraint v_t for $t \in [k+1, d-1]$ implies that the total volume of vertices

in L_t^m (note, $\frac{|L_t^m|}{|L_t|} = \frac{d-1}{d}$) plus the total volume of vertices in L_t^e (note, $\frac{|L_t^e|}{|L_t|} = \frac{1}{d}$) does not exceed the total volume assigned in those phases. Constraints m_t for $t \in [k, d-1]$ preserve the monotonicity property, i.e., the average total load of vertices in $L_{t,s}^m$ is higher than the average total load of vertices in $L_{e,s}^m$, by the groups' definition. Constraints c_t for $t \in [k, d-1]$ preserve the total volume of L_t^m after phase t as the total volume of L_{t+1} prior to phase $t+1$. Finally, the objective function captures the fractional volume assigned on L .

4.2 The dual linear program

The dual of the linear program is:

$$\begin{aligned} \min k \cdot v_k + \sum_{t=k}^{d-1} v_t + |L_d| \\ (d-k) \cdot v_k - m_k + d \cdot c_k \geq n \cdot (1 - \frac{k}{d}) & \quad (x_k^m) \\ k \cdot v_k + m_k \geq \frac{n \cdot k}{d} & \quad (x_k^e) \\ (d-1) \cdot v_t - m_t + d \cdot c_t \geq n \cdot (1 - \frac{k}{d}) \cdot (1 - \frac{1}{d})^{t-k} & \quad \forall t \in [k+1, d-1], \quad (x_t^m) \\ v_t + m_t \geq \frac{n}{d} \cdot (1 - \frac{k}{d}) \cdot (1 - \frac{1}{d})^{t-k-1} & \quad \forall t \in [k+1, d-1], \quad (x_t^m) \\ -m_t - (d-1) \cdot c_{t-1} + d \cdot c_t \geq 0 & \quad \forall t \in [k+1, d-1], \quad (u_t^m) \\ m_t - c_{t-1} \geq 0 & \quad \forall t \in [k+1, d-1], \quad (u_t^e) \\ -d \cdot c_{d-1} \geq -n \cdot (1 - \frac{k}{d}) \cdot (1 - \frac{1}{d})^{d-k-1} & \quad (u_d^m) \end{aligned}$$

4.3 Constructing the dual solution

By assuming the constraints are tight, we have by constraint (u_d^m) : $c_{d-1} = \frac{n}{d} \cdot (1 - \frac{1}{d})^{d-k-1}$, by summing constraints (u_t^m) and (u_t^e) for all $t \in [k+1, d-1]$: $c_t = c_{t-1}$, and, therefore,

$$c_t = \frac{n}{d} \cdot (1 - \frac{1}{d})^{d-k-1}, \text{ for all } t \in [k, d-1].$$

By summing constraints (x_k^m) and (x_k^e) for all $t \in [k+1, d-1]$: $d \cdot v_k + d \cdot c_k = n$, therefore:

$$v_k = n \cdot \frac{1 - d \cdot c_k}{d} = \frac{n}{d} \cdot \left(1 - (1 - \frac{k}{d}) \cdot (1 - \frac{1}{d})^{d-k-1}\right).$$

By summing constraints (x_t^m) and (x_t^e) for all $t \in [k+1, d-1]$, we have: $d \cdot v_t + d \cdot c_t = n \cdot (1 - \frac{k}{d}) \cdot (1 - \frac{1}{d})^{t-k-1}$, therefore:

$$v_t = \frac{n}{d} \cdot (1 - \frac{k}{d}) \cdot (1 - \frac{1}{d})^{d-k-1} \left((1 - \frac{1}{d})^{t-d} - 1 \right), \text{ for all } t \in [k+1, d-1].$$

Finally, by constraint x_k^e , $m_k = \frac{n \cdot k}{d} - k \cdot v_k$, and by constraint x_t^m , $m_t = \frac{n}{d} \cdot (1 - \frac{k}{d}) \cdot (1 - \frac{1}{d})^{t-k-1} - v_t$. It is easy to verify that the dual is feasible. The value of the dual objective function:

$$\begin{aligned} k \cdot v_k + \sum_{t=k+1}^{d-1} v_t + |L_d| &= \frac{n \cdot k}{d} \cdot \left(1 - (1 - \frac{k}{d}) \cdot (1 - \frac{1}{d})^{d-k-1}\right) \\ &\quad + \sum_{t=k+1}^{d-1} \frac{n}{d} \cdot (1 - \frac{k}{d}) \cdot (1 - \frac{1}{d})^{d-k-1} \left((1 - \frac{1}{d})^{t-d} - 1 \right) \\ &\quad + n \cdot (1 - \frac{k}{d}) \cdot (1 - \frac{1}{d})^{d-k-1} \\ &= n \cdot \tau(k, d) \end{aligned}$$

Therefore, there is a feasible solution for the dual, with the value $n \cdot \tau(k, d)$. By weak duality, and since the primal captures the objective of the online algorithm, any online algorithm will fractionally match at most $n \cdot \tau(k, d)$ vertices. Finally, since by our construction, the optimal value is n , any online is at most $\tau(k, d)$ competitive as required. See discussion in [3] for applying the lower bound also on randomized algorithms. \blacktriangleleft

5 Allocation problem

We improve the competitive ratio for the *allocation problem* on (k, d) -graphs for $k \leq d$.

The allocation problem

In the allocation problem, a seller is interested in selling products to a group of buyers $L = [n]$, where buyer $i \in L$ has a budget $B(i)$, and R is a set of products the seller introduces one by one. Each product $j \in R$ has a fixed price $b(j)$. Upon the arrival of a product, the buyers announce to the seller whether they are interested in buying the current product for the set price. The seller then decides (instantly) which of the interested buyers to sell the product to. Using the AdWords formulation, we have $b(i, j) \in \{0, b(j)\}$ for all i .

For $k \geq d$, there exists a trivial online fractional solution. Assume $k \leq d$ and let $\alpha = k/d$ and $\tau(\alpha) = 1 - (1 - \alpha) \cdot e^{\alpha-1}$, we will show that there exists a $\tau(\alpha)$ -competitive fractional online algorithm.

► **Theorem 7.** *For the fractional online allocation problem in (k, d) -graphs, where $\alpha = k/d$, there exists a $\tau(\alpha)$ -competitive algorithm.*

Note that the upper bound of Section 4 also holds for the allocation problem; hence, the result is tight. The algorithm and analysis are similar to the upper bound in Section 4, except we use a smooth potential function $f^{(\alpha)}$.

Let $\alpha = k/d$, $\tau(\alpha) = 1 - (1 - \alpha) \cdot e^{\alpha-1}$, and for a fixed alpha, set $\beta = 1/\tau(\alpha)$, and define $g(x) = 1 + \beta \cdot (\exp(x - 1) - 1)$, and $f^{(\alpha)}$,

$$f^{(\alpha)}(x) = \begin{cases} g(x) & \text{for } x \in [\alpha, 1] \\ \frac{x \cdot g(\alpha)}{\alpha} & \text{for } x \in [0, \alpha], \end{cases}$$

► **Lemma 8.** *For $f = f^{(\alpha)}$:*

- $f(0) = 0$, $f(1) = 1$.
- f and f' are continuous functions in the segment $[0, 1]$, and f, f' are monotone non-decreasing functions.
- For $x \in [\alpha, 1]$ we have $1 - f(x) + f'(x) = \beta$

Proof. Clearly, $f(x), f'(x)$ are continuous functions in the segments $[0, \alpha], (\alpha, 1]$. In addition, $f_-(\alpha) = g(\alpha) = f_+(\alpha)$, so f is a continuous function and

$$\begin{aligned} f'_+(\alpha) &= g'(\alpha) = \beta \cdot \exp(\alpha - 1) \\ &= \frac{\beta \cdot \alpha \exp(\alpha - 1)}{\alpha} = \frac{\beta \cdot (1/\beta - 1 + \exp(\alpha - 1))}{\alpha} = \frac{1 + \beta \cdot (\exp(\alpha - 1) - 1)}{\alpha} \\ &= \frac{g(\alpha)}{\alpha} = f'_-(\alpha), \end{aligned}$$

therefore, f' is continuous as well. And $f'(x) > 0$ for $x \in [0, 1]$. Moreover, $f''(x) = 0$ for $x \in [0, \alpha]$ and $f''(x) = \exp(x - 1)/\beta \geq 0$; hence, f' is monotone non-decreasing. $f(0) = 0 \cdot g(\alpha)/\alpha = 0$, and $f(1) = g(1) = 1 + \beta \cdot (\exp(1 - 1) - 1) = 1$. Finally, for $x \in [\alpha, 1]$

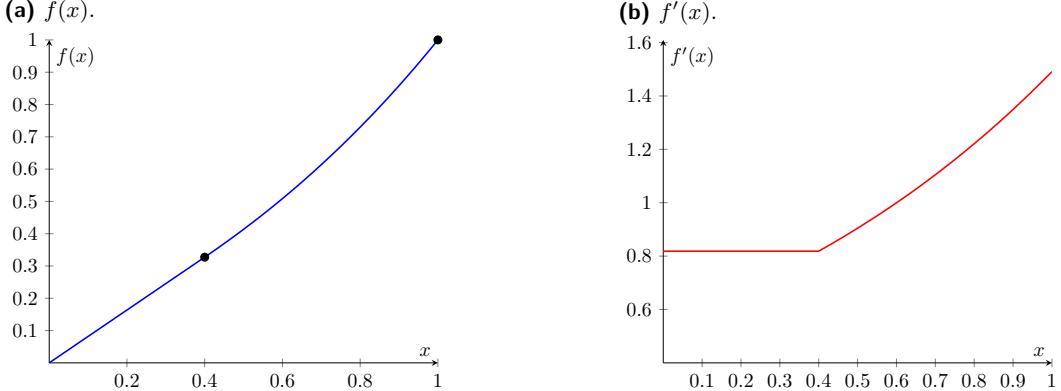


Figure 4 The functions $f^{(\alpha)}$ and $f'^{(\alpha)}$, for $\alpha = 0.4$.

$$\begin{aligned} 1 - f(x) + f'(x) &= 1 - g(x) - g'(x) \\ &= 1 - (1 + \beta \cdot (\exp(x-1) - 1)) + \beta \cdot \exp(x-1) = \beta. \end{aligned}$$

5.1 The Two-Step-Water-Level Allocation algorithm

For $i \in L$, let $N(i)$ be the set of products that buyer i is interested in. Let $T(i) = \frac{\sum_{j \in N(i)} b(j)}{B(i)}$ be the ratio between the sum of prices in which buyer i is interested, to its budget, and let $G(i) = \sum_{j \in N(i)} b(j) \cdot x_{i,j}$ be the total gain of bidder i , the level of buyer $i \in L$, defined as $\ell(i) = \frac{G(i)}{B(i)}$. Accordingly, we define the degree and level after the arrival of product j , as $T^j(i) = \sum_{t \in N(i), t \leq j} b(t)$, and $\ell^j(i) = \frac{G^j(i)}{B(i)}$, where $G^j(i) = \sum_{t \in N(i), t \leq j} b(t) \cdot x_{i,t}$.

Algorithm 2 The Two-Step-Water-Level Allocation algorithm.

Upon the arrival of a new product $j \in R$:

1. For each buyer $i \in N(j)$, such that $T^j(i) \leq k$
Increase (if necessary) $G^j(i)$, the gain of buyer i , to $T^j(i)/d$
2. Using the leftover volume of product j :
Increase the gain of bidders in $N(j)$ using the water-level algorithm (according to the buyer's level)
3. Update the dual variables, let $\ell(j) = \min_{i \in N(j)} \ell^j(i)$ and $\hat{\ell}(j) = \max(k/d, \ell(j))$.
Set $y_i = f^{(\alpha)}(\ell(i))$, for $i \in N(j)$
Set $z_j = 1 - f^{(\alpha)}(\hat{\ell}(j))$

5.2 The algorithm's feasibility

First, we will prove that for a proper $f^{(\alpha)}$ (for brevity's sake, we use f for $f^{(\alpha)}$ for the rest of the section), the primal and the dual solutions of **Two-Step-Water-Level Allocation algorithm** (TSWLA) are feasible.

► **Lemma 9.** Given a function $f : [0, 1] \rightarrow [0, 1]$, where f is monotone, non-decreasing the primal and dual solutions of the TSWLA algorithm are feasible at the end of the algorithm.

Proof. First, we will show that the algorithm does not over-allocate product j in step 1. We assume that the algorithm is feasible until the arrival of product j , and we bound the value of $x_{i,j}$ after step 1, and we will show that $x_{i,j} \leq 1/d$, for all $i \in N(j)$. For $i \in N(j)$, by our assumption that the algorithm is feasible until the arrival of product j , its level before the arrival of j is at least $\frac{T^j(i) - b(j)/B(i)}{d}$, by rearranging the terms $x_{i,j} \leq \frac{1}{d}$. Second, $\sum_{i \in N(j)} x_{i,j} \leq \frac{|N(j)|}{d} \leq 1$, by d 's definition. In the second step, the water level algorithm never increases the level above one, and, therefore, $G(i) \leq B(i)$ for all $i \in L$. We will show that for any monotone, non-decreasing function $f : [0, 1] \rightarrow [0, 1]$, the dual is feasible at the end of the algorithm's run. We have for all $i \in L$, $\ell(i) \geq \frac{\min(T(i), k)}{d} \geq k/d$, where the last inequality is by k 's definition. The dual constraints: If $\ell(j) \leq k/d$, for $i \in N(j)$, we have

$$b(j) \cdot y_i + z_j = b(j) \cdot f(\ell(i)) + b(j) \cdot (1 - f(k/d)) \geq b(j),$$

since $\ell(i) \geq k/d$ for all $i \in L$ at the end of the algorithm and f is monotone, non-decreasing. If $\ell(j) > k/d$, we have,

$$b(j) \cdot y_i + z_j \geq b(j) \cdot f(\ell^j(i)) + b(j) \cdot (1 - f(\hat{\ell}(j))) \geq b(j),$$

where the first inequality is because $\ell(i)$ only increases, and f is monotone, non-decreasing. ◀

5.3 Algorithm's competitive ratio

We are now able to complete the proof of Theorem 7. By proving that the value of the Primal is at least the $\tau(\alpha) = \frac{1}{\beta}$ times the value of the dual solution, then, by weak duality, we will conclude the TSWLA is $\tau(\alpha)$ -competitive as required.

We prove it by bounding the ratio of the increment of the primal and the dual for every arrival. We denote $\Delta P(\Delta D)$ to denote the increment for the primal (dual) variable. Moreover, $\Delta D = \Delta L + \Delta R$, where $\Delta L(\Delta R)$ refers to the increment from the left side (right side). Now, we have the following key lemma:

► **Lemma 10.** For every arrival of a product $j \in R$, we have $\Delta D \leq \Delta P \cdot \beta$.

Proof. At every step t , we have $\Delta R = z_t$, $\Delta L = \sum_{i \in N(j)} \Delta L_i$, where $\Delta L_i = B(i) \cdot (y_i^j - y_i^{j-1})$

We note that, $\frac{d\Delta P_i}{dx_{i,j}} = b(j)$. Therefore, $\Delta P = b(j) \cdot \sum_{i \in N(j)} x_{i,j}$.

$$\frac{d\Delta L_i}{dx_{i,j}} = B(i) \cdot \frac{dy_i}{dx_{i,j}} = B(i) \cdot \frac{b(j)}{B(i)} \cdot f'(\ell(i)) \leq b(j) \cdot f'(\ell^j(i)) = b(j) \cdot f'(\hat{\ell}(j))$$

where the inequality is because f' is a monotone non-decreasing function, and the last equality is since $f'(x) = f'(\alpha)$ for $x \in [0, \alpha]$.

Therefore, we have $\Delta L \leq b(j) \cdot f'(\hat{\ell}(j)) \sum_{i \in N(j)} x_{i,j}$

Case 1. $\sum_i x_{i,j} < 1$. In this case, $\ell(j) = 1$, $\Delta R = z_j = 1 - f(1) = 0$, and we have

$$\Delta D = \Delta L + \Delta R \leq b(j) \cdot f'(1) \cdot \sum_i x_{i,j} = (1 - f(1) + f'(1)) \cdot \Delta P = \beta \cdot \Delta P$$

Case 2. $\sum_i x_{i,j} = 1$, $\ell(j) > k/d$. We have $\Delta R = z_j = b(j) \cdot (1 - f(\ell(j)))$ and $\Delta P = b(j)$.

$$\Delta D = \Delta L + \Delta R \leq b(j) \cdot f'(\ell(j)) + b(j) \cdot (1 - f(\ell(j))) = b(j) \cdot (1 - f(\ell(j)) + f'(\ell(j))) = \beta \cdot \Delta P$$

Case 3. $\sum_i x_{i,j} = 1$, $\ell(j) \leq k/d$. Again, we have $\Delta P = b(j)$. We know that $\Delta R = z_j = b(u) \cdot (1 - f(k/d))$. Note that $f'(x) = f'(k/d)$ for $x \in [0, k/d]$.

$$\Delta D = \Delta L + \Delta R \leq b(j) \cdot f'(\ell(j)) + b(j) \cdot (1 - f(k/d)) = b(j) \cdot (1 - f(k/d) + f'(k/d)) = \beta \cdot \Delta P. \blacksquare$$

6 Upper Bounds for the Adwords problem

In this section, we prove that $\tau(k, d)$ competitiveness is impossible for the Adwords problem.

► **Theorem 11.** *For fractional AdWords in (k, d) -graphs, there exists k, d and fixed $\epsilon > 0$, such that no online algorithm can be better than $(\tau(k, d) - \epsilon)$ -competitive.*

The hard instance is composed of several scenarios. The intuition behind the scenario is that if for a certain j , $b(i_1, j) > b(i_2, j)$, then, on the one hand, the online algorithm should allocate the item to i_1 in order not to lose its higher price, and, on the other hand, by doing so the optimal allocation can use item j to i_2 . We amplify this example and prove an improved upper bound for $\alpha = 1/2$ ($k = d/2$) and $B(i) = 1$ for all $i \in L$. First, we bound the gain of a subset of the bidders after several sequence steps.

6.1 Bounding the gain of a subset of the bidders

Let $Q \subseteq L$ be a subset of vertices. After several sequence steps, we bound the total online gain that can be extracted from Q , where the degree of $i \in Q$ is already k , and the adversary determined how many bidders from Q have already exhausted their budget in previous steps. We compute an upper bound on the total gain of an online algorithm on Q as a function of the average online gain on Q in previous steps by defining the rest of the sequence for this subset Q and bounding the total gain using a linear program. The rest of the sequence for this Q is as the second part of Section 4. Specifically, it defines Q_{k+1} as a subset of Q in which the current gain in OPT is 0. In phase $t \in [k+1, d-1]$, the adversary divides Q_t into $|Q_t|/d$ groups and introduces for each group's bidders a product with an equal bid value of 1. The adversary defines Q_{t+1} by dropping the lowest gain bidder from each group. In the last phase, for each bidder $i \in Q_d$, the adversary introduces a product with a bid of 1.

Formally, given such Q and the decomposition of Q into Q_k^e, Q_k^m where the average gain in Q_k^e is, at most, the average gain of Q_k^m . For $t \in [k+1, d]$, the adversarial would define at each phase $t \in [k+1, d]$, $Q_t \subseteq Q$, where $Q_{k+1} = Q_k^m$. The adversary groups each Q_t into $|Q_t|/d$ disjoint groups, each of size d , denoted as $Q_{t,s}$, for $s \in [|Q_t|/d]$. In phase $t \in [k+1, d-1]$, $|Q_t|/d$ products arrive and the s 'th ($s \in [|Q_t|/d]$) bid $b(i, s) = 1$ to $i \in Q_{t,s}$ (and 0 otherwise). In phase $t \in [k+1, d-1]$, $|Q_t|/d$ products arrive and the s 'th ($s \in [|Q_t|/d]$) product bids are $b(i, s) = 1$ to all the bidders in group $i \in Q_{t,s}$. Q_{t+1} for $t \in [k+1, d-1]$ is set as follows: For $s \in \frac{|Q_t|}{d}$, let $Q_{k,s}^e \subset Q_{k,s}$ be the subset containing the lowest gain bidder in $Q_{t,s}$ after phase t and $Q_{t,s}^m = Q_{t,s} \setminus Q_{k,s}^e$. We define $Q_{t+1} = \bigcup_{s=1}^{|Q_t|/d} Q_{t,s}^m$. note that, $|Q_{t+1}| = (1 - 1/d)|Q_t|$ for $t \in [k+1, d-1]$. Let G_Q be the total gain of Q vertices.

Bounding G_Q as a function of previous phases. Next, we bound the total gain of G_Q , given that after phase k , a decomposition of Q into Q_k^e, Q_k^m exists (i.e., $Q = Q_k^e \cup Q_k^m$) such that x_k^m (x_k^e) is the average load on Q_k^m (Q_k^e) and $x_k^e \leq x_k^m$ (constraint \tilde{m}_k). Assuming that the total gain of Q until (not including) step $k+1$ is G_Q^k , we have $\frac{|Q_k^m|}{|Q|} \cdot x_k^m + \frac{|Q_k^e|}{|Q|} \cdot x_k^e \leq \frac{G_Q^k}{|Q|}$, constraint \tilde{v}_k . We will define $Q_{k+1} = Q_k^m$, and accordingly, we have $d \cdot x_k^m \leq (d-1) \cdot u_{k+1}^m + u_{k+1}^e$ (constraint \tilde{n}_k). Denote $V^Q = \frac{G_Q^k}{|Q|}$, and $r^Q = \frac{|Q_k^m|}{|Q|}$ the following Linear program, will bound $G(V^Q, r^Q)$ the total gain that can be extracted from Q .

$G(V^Q, r^Q) = \max |Q| \cdot V^Q + \sum_{t=k+1}^{d-1} \frac{|Q_t|}{d} \cdot (x_t^e + (d-1)x_t^m) + |Q_d| \cdot (1 - u_d^m)$,
such that:

$$\begin{aligned} r^Q \cdot x_k^m + (1 - r^Q) \cdot x_k^e &\leq V^Q & (\tilde{v}_k) \\ (d-1) \cdot x_t^m + x_t^e &\leq 1 & \forall t \in [k+1, d-1], \quad (v_t) \\ x_k^e &\leq x_k^m & (\tilde{m}_k) \\ x_t^e + u_t^e &\leq x_t^m + u_t^m & \forall t \in [k+1, d-1], \quad (m_t) \\ d \cdot x_k^m &\leq (d-1) \cdot u_{k+1}^m + u_{k+1}^e & (\tilde{n}_k) \\ d \cdot (x_t^m + u_t^m) &\leq (d-1) \cdot u_{t+1}^m + u_{t+1}^e & \forall t \in [k+1, d-2], \quad (c_t) \\ d \cdot (x_{d-1}^m + u_{d-1}^m) &\leq d \cdot u_d^m & (c_{d-1}) \\ x_t^m, u_t^m, x_t^e, u_t^e &\geq 0 & \forall t \in [k, d] \end{aligned}$$

6.2 The Scenarios

Let $L = U \cup D$, such that $|U| = |D| = n$. We divide $U(D)$ into n/k disjoint groups $U_{k,s}(D_{k,s})$ such that $|U_{k,s}| = |D_{k,s}| = k$, for $s \in [n/k]$.

Phase 0. For each $s \in [n/k]$, introduce a product j , such that $b(i, j) = 0.26 \cdot d$ for $i \in U_{k,s}$ and $b(i, j) = 0.24 \cdot d$ for $j \in D_{k,s}$. Any online algorithm must determine $\gamma = \sum_{i \in U_{k,s}, j} x_{i,j}/n$, the average portion of the items of phase 0 assigned to U . Next, the adversary introduces products of phase k (phases $[k-1]$ are empty).

First Scenario. For each $s \in [n/(2 \cdot k)]$, the adversary introduces two products j_1, j_2 , such that $b(i, j_1) = 0.24 \cdot d$ for $i \in U_{k,2 \cdot s-1} \cup U_{k,2 \cdot s}$ and $b(i, j_2) = 0.26 \cdot d$ for $i \in D_{k,2 \cdot s-1} \cup D_{k,2 \cdot s}$.

Second Scenario. For each $s \in [n/k]$, the adversary introduces a product j , such that $b(i, j) = 0.24 \cdot d$ for $i \in U_{k,s}$ and $b(i, j) = 0.26 \cdot d$ for $j \in D_{k,s}$. Any online algorithm must determine $\delta = \sum_{i \in U_{k,s}, j} x_{i,j}/n$, the average portion of the items of phase k of the second scenario assigned to U . Note that, in both scenarios, the current degree of each vertex $i \in L$ is k . Using the values γ, δ , we bound the average gain per bidder of $U(D)$ up to phase k for scenario o , denoted as $V^{U(o)}(V^{D(o)})$. For the first scenario:

$$\begin{aligned} V^{U(1)} &= \frac{1}{n} \cdot \left(\frac{n}{k} \cdot \gamma \cdot 0.26 \cdot d + \frac{n}{d} \cdot 0.24 \cdot d \right) = 0.52 \cdot \gamma + 0.24 \\ V^{D(1)} &= \frac{1}{n} \cdot \left(\frac{n}{k} \cdot (1-\gamma) \cdot d \cdot 0.24 \cdot d + \frac{n}{d} \cdot 0.26 \cdot d \right) = 0.74 - 0.48 \cdot \gamma \end{aligned}$$

For the second scenario:

$$\begin{aligned} V^{U(2)} &= \frac{1}{n} \cdot \left(\frac{n}{k} \cdot \gamma \cdot 0.26 \cdot d + \frac{n}{k} \cdot \delta \cdot 0.24 \cdot d \right) = 0.52 \cdot \gamma + 0.48 \cdot \delta \\ V^{D(2)} &= \frac{1}{n} \cdot \left(\frac{n}{k} \cdot (1-\gamma) \cdot d \cdot 0.24 \cdot d + \frac{n}{k} \cdot (1-\delta) \cdot 0.26 \cdot d \right) = 1 - 0.48 \cdot \gamma - 0.52 \cdot \delta \end{aligned}$$

Similarly, we denote $\gamma^{\text{OPT}}, \delta^{\text{OPT}}$ as the corresponding values for the optimal allocation. After setting those values (for a certain case), the gain of OPT on this subset would be determined. Then, the adversary can omit bidders in the corresponding subset, i.e., determining r^Q for $Q \in U, D$.

Continuing the first scenario. In the first scenario, the adversary sets $\gamma^{\text{OPT}} = 0$ (i.e., uses the products of phase 0 to increase the gain of vertices only in D). Taking into account also the products of phase k of the first scenario, it omits the $r^U = 0.24$ portion from U and the $r^D = 0.74$ portion from D , and continues the sequence for U, D separately using Subsection 6.1 construction.

Formally, for $s \in [n/k]$, let $U_{k,j}^e \subset U_{k,s}$ be the subset of the $(r^U \cdot k)$ -lowest gain vertices in $U_{k,s}$ after phase k and $U_{k,s}^m = U_{k,s} \setminus U_{k,j}^e$ (the subset of the $((1 - r^U) \cdot k)$ -highest load vertices in $U_{k,j}$), and define $U_{k+1} = \cup_s U_{k,s}^m$. Accordingly, for $s \in [n/k]$, let $D_{k,s}^e \subset D_{k,s}$ be the subset of the $(r^D \cdot k)$ -lowest gain vertices in $D_{k,s}$ after phase k and $D_{k,s}^m = D_{k,s} \setminus Q_{k,j}^e$ (the subset of the $((1 - r^D) \cdot k)$ -highest load vertices in $Q_{k,j}$). We continue the scenario separately for U and D using Subsection 6.1 construction. We bound the total gain of this scenario using the LP $G(V^Q, r^Q)$. Specifically, the total gain is at most (as a function of γ):

$$G(V^U, r^U) + G(V^D, r^D) = G(0.52 \cdot \gamma + 0.24, 0.24) + G(0.74 - 0.48 \cdot \gamma, 0.74).$$

Continuing the second scenario. In the second scenario, the adversary uses $\gamma^{\text{OPT}} = 1$, $\delta^{\text{OPT}} = 0$ (i.e., uses the products of phase 0 to increase the gain of vertices only in U and the products of the second scenario of phase k to increase the gain of vertices only in D). In this case, it omits the $r^U = 0.52$ portion from U and the $r^D = 0.52$ portion from D . Similarly to the previous case, we have:

$$G(V^U, r^U) + G(V^D, r^D) = G(0.52 \cdot \gamma + 0.48 \cdot \delta, 0.52) + G(1 - 0.48 \cdot \gamma - 0.52 \cdot \delta, 0.52).$$

By our construction definition, in all scenarios, the optimal value is $2 \cdot n$. Therefore, in order to bound c , we have the following LP

$$\max c$$

$$\text{such that: } G(0.52 \cdot \gamma + 0.24, 0.24) + G(0.74 - 0.48 \cdot \gamma, 0.74) \geq 2 \cdot n \cdot c$$

$$G(0.52 \cdot \gamma + 0.48 \cdot \delta, 0.52) + G(1 - 0.48 \cdot \gamma - 0.52 \cdot \delta, 0.52) \geq 2 \cdot n \cdot c$$

By solving it numerically, we have for $d = 100$, $k = 50$, we have:

$$c \leq 0.69485 < \tau(0.5) \approx 0.6967.$$

References

- 1 Susanne Albers and Sebastian Schubert. Online ad allocation in bounded-degree graphs. In *Web and Internet Economics: 18th International Conference, WINE 2022, Troy, NY, USA, December 12–15, 2022, Proceedings*, pages 60–77. Springer, 2022.
- 2 Susanne Albers and Sebastian Schubert. Tight bounds for online matching in bounded-degree graphs with vertex capacities. In *ESA*, volume 244 of *LIPICS*, pages 4:1–4:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.
- 3 Yossi Azar, Ilan Reuven Cohen, and Alan Roytman. Online lower bounds via duality. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1038–1050. SIAM, 2017.
- 4 Bahman Bahmani and Michael Kapralov. Improved bounds for online stochastic matching. In *Algorithms–ESA 2010: 18th Annual European Symposium, Liverpool, UK, September 6–8, 2010. Proceedings, Part I 18*, pages 170–181. Springer, 2010.
- 5 Niv Buchbinder, Kamal Jain, and Joseph Naor. Online primal-dual algorithms for maximizing ad-auctions revenue. In *Algorithms–ESA 2007: 15th Annual European Symposium, Eilat, Israel, October 8–10, 2007. Proceedings 15*, pages 253–264. Springer, 2007.

- 6 Niv Buchbinder, Joseph Seffi Naor, et al. The design of competitive online algorithms via a primal-dual approach. *Foundations and Trends® in Theoretical Computer Science*, 3(2–3):93–263, 2009.
- 7 Jon Feldman, Aranyak Mehta, Vahab Mirrokni, and Shan Muthukrishnan. Online stochastic matching: Beating 1-1/e. In *2009 50th Annual IEEE Symposium on Foundations of Computer Science*, pages 117–126. IEEE, 2009.
- 8 Bala Kalyanasundaram and Kirk R. Pruhs. An optimal deterministic algorithm for online b-matching. *Theoretical Computer Science*, 233(1-2):319–325, 2000.
- 9 Chinmay Karande, Aranyak Mehta, and Pushkar Tripathi. Online bipartite matching with unknown distributions. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 587–596, 2011.
- 10 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.
- 11 Mohammad Mahdian and Qiqi Yan. Online bipartite matching with random arrivals: an approach based on strongly factor-revealing lps. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 597–606, 2011.
- 12 Vahideh H Manshadi, Shayan Oveis Gharan, and Amin Saberi. Online stochastic matching: Online actions based on offline statistics. *Mathematics of Operations Research*, 37(4):559–573, 2012.
- 13 Aranyak Mehta, Amin Saberi, Umesh Vazirani, and Vijay Vazirani. Adwords and generalized online matching. *Journal of the ACM (JACM)*, 54(5):22–es, 2007.
- 14 Joseph Naor and David Wajc. Near-optimum online ad allocation for targeted advertising. *ACM Transactions on Economics and Computation (TEAC)*, 6(3-4):1–20, 2018.