Tight Bounds for Online Matching in Bounded-Degree Graphs with Vertex Capacities

Susanne Albers
Department of Computer Science, Technische Universität München, Germany

Sebastian Schubert[1]
Department of Computer Science, Technische Universität München, Germany

Abstract

We study the b-matching problem in bipartite graphs $G = (S, R, E)$. Each vertex $s \in S$ is a server with individual capacity b_s. The vertices $r \in R$ are requests that arrive online and must be assigned instantly to an eligible server. The goal is to maximize the size of the constructed matching. We assume that G is a (k, d)-graph [19], where k specifies a lower bound on the degree of each server and d is an upper bound on the degree of each request. This setting models matching problems in timely applications.

We present tight upper and lower bounds on the performance of deterministic online algorithms. In particular, we develop a new online algorithm via a primal-dual analysis. The optimal competitive ratio tends to 1, for arbitrary $k \geq d$, as the server capacities increase. Hence, nearly optimal solutions can be computed online. Our 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.

Our bounds improve the previous best competitive ratios. The asymptotic competitiveness of 1 is a significant improvement over the previous factor of $1 - 1/e^{k/d}$, for the interesting range where $k/d \geq 1$ is small. Recall that $1 - 1/e \approx 0.63$. Matching problems that admit a competitive ratio arbitrarily close to 1 are rare. Prior results rely on randomization or probabilistic input models.

2012 ACM Subject Classification Theory of computation → Online algorithms

Keywords and phrases online algorithms, deterministic algorithms, primal-dual analysis, b-matching, bounded-degree graph, variable vertex capacities, unweighted matching, vertex-weighted matching

Digital Object Identifier 10.4230/LIPIcs.ESA.2022.4

Related Version *Full Version*: `https://arXiv.org/abs/2206.15336`

1 Introduction

Maximum matching is a fundamental problem in computer science. In a seminal paper Karp, Vazirani and Vazirani [15] introduced online matching in bipartite graphs $G = (S \cup R, E)$. The vertices of S are known in advance, while the vertices of R (requests) arrive one by one and must be matched immediately to an eligible partner. The b-matching problem is a generalization where the vertices of S (servers) have capacities and may be matched multiple times, see e.g. [14]. Online bipartite matching and capacitated extensions have received tremendous research interest over the past 30 years. In this paper we study the b-matching problem in bounded-degree graphs, defined in [19]. We assume that there is a lower bound on the degree of each server $s \in S$, meaning that there is a certain demand for each server. Furthermore we assume that there is an upper bound on the degree of each $r \in R$, i.e. each request can only be assigned to a subset of the servers. This setting models matching problems in many timely applications, as we will describe below.

[1] Corresponding author

More formally, we investigate the following problem. Again, let $G = (S \cup R, E)$ be a bipartite graph, where the vertices of S are servers and the vertices of R are requests. The set S is known in advance. Each server $s \in S$ has an individual capacity $b_s \in \mathbb{N}$, indicating that the server can be matched with up to b_s requests. The vertices of R arrive online, one by one. Whenever a new request $r \in R$ arrives, its incident edges are revealed. The request has to be matched immediately and irrevocably to an eligible server, provided that there is one. The goal is to maximize the number of matching edges. We will also examine the *vertex-weighted* problem extension, where additionally each server $s \in S$ has a weight w_s and the value of every matching edge incident to s is multiplied by w_s. Now the goal is to maximize the total weight of the constructed matching.

We assume that G is a (k, d)-graph, defined by Naor and Wajc [19], where k and d are positive integers. Each server $s \in S$ has a degree $d(s) \geq k \cdot b_s$. Each request $r \in R$ has a degree $d(r) \leq d$. Naor and Wajc [19] defined these graphs for the general AdWords problem. Note that the inequality $d(s) \geq k \cdot b_s$ expresses a degree bound in terms of the server capacity. This is essential. As we shall see, the performance of algorithms depends on the degrees $d(s)$ as a function of b_s. A degree bound independent of b_s is vacuous for larger b_s. Also, a company operating a high-capacity server expects the server to be attractive and a potential host for a large number of requests. Otherwise it might be beneficial to reduce the server capacity.

The best results will be obtained if $k \geq d$. In this case the average demand for each server slot is high, compared to the number of servers a request can be assigned to. This setting is also very relevant in applications. We will assume that $d \geq 2$. If $d = 1$, any Greedy algorithm constructs an optimal matching. We remark that (k, d)-graphs are loosely related to d-regular graphs in which each vertex has a degree of exactly d and a capacity of 1. This graph class has been studied extensively in computer science and discrete mathematics, see e.g. [5, 6, 7, 10, 20].

The b-matching problem in (k, d)-graphs models many problems in modern applications, cf. [4, 12, 19]. The following description also addresses the degree constraints.

Video content delivery, web hosting, remote data storage: Consider a collection of servers in a video content delivery network, a web hosting provider, or a remote data storage service. A sequence of clients arrives, each with a request that videos be streamed, web pages be hosted, or data be stored. Based on the servers' geographic distribution, average performance, technology used or pricing policies, each request can only be hosted at a small subset of the servers or server locations. Each server has a large capacity and is well suited to service a huge number of requests in the arriving client sequence.

Job scheduling: Consider a collection of compute servers, each with certain capabilities, located for instance in a data center. Over a time horizon jobs arrive, requesting service. Based on computing demands, expected response time, hardware and software requirements, each job can only be executed on a subset of the servers. During the given time horizon, each server can process a large number of jobs and is a suitable platform to execute very many of the incoming jobs.

AdWords and ad auctions: Consider a search engine company or digital advertising platform. There is a set of advertisers, each with a daily budget, who wish to link their ads to users of the search engine/digital platform and issue respective bids. The users arrive online and must be allocated instantly to the advertisers. Based on his search keywords, browsing history and possible profile, each user is interesting to a small set of advertisers. Each advertiser has a decent budget and targets a large population of the users. Obviously, in this application the advertisers correspond to the servers and the users are the incoming

requests. The b-matching problem models the basic setting where the bids of all advertisers are either 0 or 1. The vertex-weighted extension captures the scenario where all the bids of an advertiser $s \in S$ have a value of 0 or w_s. These base cases are also studied in recent work by Vazirani [21].

We analyze the performance of online algorithms using competitive analysis. Given an input graph G, let $\textsc{Alg}(G)$ denote the size (or weight) of the matching constructed by an online algorithm Alg. Let $\textsc{Opt}(G)$ be the corresponding value of an optimal offline algorithm Opt. Algorithm Alg is c-competitive if $\textsc{Alg}(G) \geq c \cdot \textsc{Opt}(G)$ holds, for all G. In our analyses we will focus on bipartite (k, d)-graphs G.

Related work. As mentioned above, Karp et al. [15] introduced online matching in bipartite graphs, in which every vertex has a capacity of 1. The best competitive ratio of deterministic online algorithms is equal to 1/2. Karp et al. proposed a randomized Ranking algorithm that achieves an optimal competitive ratio of $1 - 1/e \approx 0.63$. Aggarwal et al. [1] defined online vertex-weighted bipartite matching and devised a $(1 - 1/e)$-competitive algorithm.

Kalyanasundaram and Pruhs [14] investigated the b-matching problem if all servers have equal capacity, i.e. $b_s = b$ for all $s \in S$. They presented a deterministic Balance algorithm that matches a new request to an adjacent server whose current load is smallest. Balance achieves an optimal competitive ratio of $1 - 1/(1 + 1/b)^b$. As b grows, the latter expression tends from below to $1 - 1/e$. Grove et al. [12] and Chaudhuri et al. [4] studied b-matchings with a different objective. At any time an algorithm must maintain a matching between the requests that have arrived so far and the servers. The goal is to minimize the total number of switches, reassigning requests to different servers.

The AdWords problem was formally defined by Mehta et al. [18]. They presented a deterministic online algorithm that achieves a competitive ratio of $1-1/e$, under the *small-bids assumption* where the bids are small compared to the advertisers' budgets. No randomized algorithm can obtain a better competitive factor. Buchbinder et al. [3] examined a setting where the degree of each incoming user is upper bounded by d and gave an algorithm with a competitive ratio of nearly $1 - (1 - 1/d)^d$. Azar et al. [2] showed that this ratio is best possible, also for randomized algorithms. The expression $1 - (1 - 1/d)^d$ is always greater than $1 - 1/e$ but approaches the latter value as d increases.

The class of (k, d)-graphs was defined by Naor and Wajc [19], who studied online bipartite matching and the AdWords problem. They proposed an algorithm HighDegree that matches a new request to an available neighbor of highest current degree. Naor and Wajc proved that HighDegree and generalizations attain a competitive factor of $1 - (1 - 1/d)^k$. This ratio holds for online bipartite matching and the vertex-weighted extension, where all vertices have a capacity of 1. Furthermore, it holds for AdWords with equal bids per bidder. For AdWords with arbitrary bids, the ratio is $(1 - R_{\max})(1 - (1 - 1/d)^k)$, where $R_{\max}$ is the maximum ratio between the bid of any bidder and its total budget. Naor and Wajc showed that no deterministic online algorithm for bipartite matching can achieve a competitive ratio greater than $1 - (1 - 1/d)^k$ if $k \geq d$. For the general AdWords problem, they proved an upper bound of $(1 - R_{\max})(1 - (1 - 1/d)^{k/R_{\max}})$ if $k \geq d$. For increasing k/d, the expression $1 - (1 - 1/d)^k$ tends to 1. For $k \approx d$ increasing, it approaches again $1 - 1/e$.

Cohen and Wajc [5] studied online bipartite matching in d-regular graphs and developed a randomized algorithm with a competitive ratio of $1 - O(\sqrt{\log d/d})$, which tends to 1 as d increases.

Online bipartite matching and the AdWords problem have also been examined in stochastic input models. A random permutation of the vertices of R may arrive. Alternatively, the vertices of R are drawn i.i.d. from a known or unknown distribution. For online bipartite

matching, the best online algorithms currently known achieve competitive ratios of 0.696 and 0.706 [13, 16]. The best possible performance ratios are upper bounded by 0.823 [17], and hence bounded away from 1. For AdWords, $(1-\varepsilon)$-competitive algorithms are known, based on the small-bids assumption [8, 9].

Our contributions. We present a comprehensive study of the b-matching problem in (k,d)-graphs. Specifically, we develop tight lower and upper bounds on the performance of deterministic online algorithms. The optimal competitive ratio tends to 1, for any choice of k and d with $k \geq d$, as the server capacities increase.

First, in Section 2 we investigate the setting that all servers have the same capacity, i.e. $b_s = b$ for all $s \in S$. We develop an optimal online algorithm WeightedAssignment via a primal-dual analysis. The resulting strategy is simple. Associated with each server of current load l and current degree δ is a value $V(l,\delta)$. An incoming request is assigned to an eligible server for which the increment $V(l,\delta+1) - V(l,\delta)$ is maximized. The values $V(l,\delta)$ can be calculated in a preprocessing step and retrieved by table lookup when the requests of R are served. The best values $V(l,\delta)$, for variable l and δ, are determined using recurrence relations. Solving them is non-trivial because two parameters are involved.

We prove that WeightedAssignment achieves a competitive ratio of c^*, where

$$c^* = 1 - \frac{1}{b}\left(\sum_{i=1}^{b} i\binom{kb}{b-i}\frac{1}{(d-1)^{b-i}}\right)\left(1-\frac{1}{d}\right)^{kb}.$$

This is a slightly complex expression, but it is exact in all terms. In Section 3 we prove that no deterministic online algorithm can attain a competitive ratio greater than c^*, for any choices of k and d that fulfill $k \geq d$.

In Section 4 we consider two generalizations. We assume that each server $s \in S$ has an individual capacity b_s and adapt WeightedAssignment. As for the competitive factor, in c^* the capacity b has to be replaced by $b_{\min} := \min_{s\in S} b_s$. The resulting competitiveness is again optimal for $k \geq d$. Furthermore, we study the vertex-weighted problem extension and again adjust our algorithm. The competitive ratios are identical to those in the unweighted setting, for uniform and variable server capacities. Our results also hold for the AdWords problem where bidders issue individual, equally valued bids.

In Section 5 we analyze the optimal competitive ratio c^*. We prove that it tends to 1, for any $k \geq d$, as b increases. Furthermore, we show that it is strictly increasing in b. The analyses are involved and make non-trivial use of Gauss hypergeometric functions.

A strength of our results is that the optimal competitiveness tends to 1, for increasing server capacities. Hence almost optimal solutions can be computed online. For the AdWords problem, high server capacities correspond to the small-bids assumption. Remarkably, in this setting near-optimal ad allocations can be computed based on structural properties of the input graph if bidders issue individual, equally valued bids. Recall that, without degree bounds, the competitive ratio for the b-matching problem tends from below to $1-1/e \approx 0.63$. The competitiveness of c^* improves upon the previous best ratio of $1-(1-1/d)^k$ [19]. The ratio c^* is equal to $1-(1-1/d)^k$ for $b=1$ and strictly increasing in b, for any $k \geq d$. Our asymptotic competitiveness of 1 is a significant improvement over $1-(1-1/d)^k \approx 1-1/e^{k/d}$, for the interesting range of small $k/d \geq 1$. For $k < d$, $1-(1-1/d)^k$ and c^* can become small. The algorithms are still $\frac{1}{2}$-competitive since they match requests whenever possible. We are aware of only two other online matching problems that admit competitive ratios arbitrarily close to 1. As mentioned above, a randomized algorithm is known for online matching in d-regular unit-capacity graphs [5]. For the general AdWords problem, respective algorithms exist if the input R is generated according to probability distributions [8, 9].

2 An optimal online algorithm

In this section we study the setting that all servers have a uniform capacity of b. We develop our algorithm WeightedAssignment. While serving requests, the algorithm maintains a value $V(l_s, \delta_s)$, for each server s with load l_s and current degree δ_s. At any point in time during the execution of the online algorithm, the load of a server s denotes the amount of matched edges incident to s, while the current degree indicates the total number of edges incident to s. In order to construct the function V and for the purpose of analysis, we formulate WeightedAssignment as a primal-dual algorithm. The primal and dual linear programs of the b-matching problem are given below. The primal variables $m(s,r)$ indicate if an edge $\{s,r\} \in E$ belongs to the matching. We have dual variables $x(s)$ and $y(r)$.

In the pseudocode of WeightedAssignment, also stated below, Line 7 is the actual matching step. A new request r is assigned to a neighboring server s for which the difference $V(l_s, \delta_s + 1) - V(l_s, \delta_s)$ is maximized. $N(r)$ is the set of adjacent servers with remaining capacity. All other instructions essentially update primal and dual variables so that a primal and a dual solution are constructed in parallel.

Observe that no dual variable $y(r)$ of any request r is ever increased by WeightedAssignment. The dual variable $x(s)$ of a server s can be increased in Lines 9 and 11. It is increased if s is matched to a neighboring request r and, importantly, $x(s)$ is also increased if this r is assigned to a different server.

$$\textbf{P: } \max \sum_{\{s,r\}\in E} m(s,r)$$
$$\text{s.t.} \sum_{r:\{s,r\}\in E} m(s,r) \le b, \ (\forall s \in S)$$
$$\sum_{s:\{s,r\}\in E} m(s,r) \le 1, \ (\forall r \in R)$$
$$m(s,r) \ge 0, \ (\forall \{s,r\} \in E)$$

$$\textbf{D: } \min \sum_{s\in S} b \cdot x(s) + \sum_{r\in R} y(r)$$
$$\text{s.t. } x(s) + y(r) \ge 1, \ (\forall \{s,r\} \in E)$$
$$x(s),\, y(r) \ge 0, \ (\forall s \in S, \forall r \in R)$$

Algorithm 1 WeightedAssignment.

```
1  Initialize x(s) = 0, y(r) = 0 and m(s,r) = 0, ∀s ∈ S and ∀r ∈ R;
2  while a new request r ∈ R arrives do
3      Let N(r) denote the set of neighbors s of r with remaining capacity;
4      if N(r) = ∅ then
5          Do not match r;
6      else
7          Match r to arg max {V(l_s, δ_s + 1) − V(l_s, δ_s) : s ∈ N(r)};
8          Update m(s,r) ← 1;
9          Set x(s) ← V(l_s + 1, δ_s + 1);
10         forall s' ≠ s ∈ N(r) do
11             Set x(s') ← V(l_s', δ_s' + 1);
12         end
13     end
14 end
```

In the analysis, we will see how the function V has to be defined so that WeightedAssignment achieves the desired competitive ratio c^*. Note that we always construct a feasible dual solution if $x(s) = 1$ holds, for all servers $s \in S$, by the end of the algorithm.

Here lies a crucial idea of the algorithm and the construction of V. We demand $V(b,\delta)=1$, for all $\delta \geq b$, and $V(l,\delta)=1$ if $\delta \geq kb$, for all $0 \leq l \leq b$. Also, $V(0,0)=0$. These constraints have two important implications.

1. The dual variable $x(s)$ of a server s with load l_s and degree δ_s is always equal to $V(l_s,\delta_s)$: Consider an incoming request r that is a neighbor of s. While $l_s < b$, it holds $N(r) \neq \emptyset$ and r is matched to some server. Lines 9 and 11 correctly update $x(s)$ with respect to the new load and degree values. If $l_s = b$, then inductively by the first constraint $x(s)=1$ and no update is necessary.
2. The constructed dual solution is feasible: Implication 1 and the second constraint ensure that $x(s)=1$ holds for all $s \in S$ by the end of the algorithm, since every server s has a degree of at least kb.

Let P and D denote the value of the primal and dual solution constructed by the algorithm, respectively. We denote a change in the value of the primal and dual solution by ΔP and ΔD, respectively. It holds that the size of the matching constructed by WeightedAssignment is exactly $|\textsc{Alg}| = P$. Moreover, by weak duality, we get that the size of the optimum matching is $|\textsc{Opt}| \leq D$. Hence, if we were able to bound $\Delta D \leq \Delta P/c$ at every step, we would obtain a competitive ratio of c.

$$\frac{|\textsc{Alg}|}{|\textsc{Opt}|} \geq \frac{P}{D} \geq \frac{P}{\frac{1}{c}\cdot P} = c\,.$$

Recall that the value of the dual solution is only increased if a request r is matched to a server s. Then, the value of the primal solution is increased by 1, while the value of the dual solution is increased by

$$\Delta D = b \cdot \Big(V(l_s+1,\delta_s+1) - V(l_s,\delta_s) + \sum_{s' \in N(r)\setminus\{s\}} V(l_{s'},\delta_{s'}+1) - V(l_{s'},\delta_{s'})\Big)\,.$$

The algorithm chooses s such that $V(l_s,\delta_s+1) - V(l_s,\delta_s) \geq V(l_{s'},\delta_{s'}+1) - V(l_{s'},\delta_{s'})$ holds for all $s' \in N(r)$. Furthermore, $|N(r)| \leq d$ implies that we can bound this increase by

$$\Delta D \leq b \cdot \Big(V(l_s+1,\delta_s+1) - V(l_s,\delta_s) + (d-1)\cdot\big(V(l_s,\delta_s+1) - V(l_s,\delta_s)\big)\Big)\,.$$

This means, that we need to determine the biggest possible constant $c^* \in (0,1]$ such that

$$b \cdot \Big(V(l+1,\delta+1) - V(l,\delta) + (d-1)\cdot\big(V(l,\delta+1) - V(l,\delta)\big)\Big) \leq \frac{1}{c^*} \tag{1}$$

holds for all $0 \leq l < b$ and all $\delta \geq l$, while still satisfying our constraints that $V(b,\cdot)=1$ and $V(\cdot,\delta')=1$ for $\delta' \geq kb$. For this, we define

$$p(l,\delta) := V(l+1,\delta+1) - V(l,\delta) \quad \text{and} \quad q(l,\delta) := V(l,\delta+1) - V(l,\delta)\,.$$

In other words, the dual variable $x(s)$ of a server s with load l and current degree δ is increased by $p(l,\delta)$, when a request is assigned to s, and increased by $q(l,\delta)$, when a neighboring request is assigned to a different server. Our constraints immediately give $p(l,\delta) = q(l,\delta) = 0$, if $l = b$ or $\delta \geq kb$. Hence, we will focus on the case $0 \leq l < b$ and $l \leq \delta < kb$ in the following. Rewriting and rearranging inequality (1) in terms of p and q yields

$$q(l,\delta) \leq \frac{1}{d-1}\left(\frac{1}{b\cdot c} - p(l,\delta)\right)\,.$$

We treat the values $p(i,i)$, for $0 \leq i < b-1$, as the variables of our optimization, since every other p and q value can then be computed based on these choices. To get comfortable with the recursions and ideas in the latter part of this section, we do the following warm-up, where we consider $V(b-1,\delta)$. It holds

$$V(b-1,\delta) = \sum_{i=0}^{b-2} p(i,i) + \sum_{j=b-1}^{\delta-1} q(b-1,j)\,.$$

Our first constraint $V(b,\delta) = 1$, for all $\delta \geq b$, implies that $p(b-1,\delta) = 1 - V(b-1,\delta)$. We do not want to waste any potential increases in our dual variables, since we want to maximize c. Thus, we will choose the maximum possible value for $q(b-1,\delta)$, which is

$$q(b-1,\delta) = \frac{1}{d-1}\left(\frac{1}{b\cdot c} - p(b-1,\delta)\right) = \frac{1}{d-1}\left(\frac{1}{b\cdot c} - 1 + V(b-1,\delta)\right)\,.$$

It follows that

$$V(b-1,\delta+1) = V(b-1,\delta) + q(b-1,\delta) = \frac{d}{d-1}V(b-1,\delta) + \frac{1}{d-1}\left(\frac{1}{b\cdot c} - 1\right)\,, \quad (2)$$

for all $b-1 \leq \delta < kb$ and with $V(b-1,b-1) = \sum_{i=0}^{b-2} p(i,i)$. To ease notation in the future, we further define $P_i := \sum_{j=0}^{i-1} p(j,j)$, for all $0 \leq i \leq b-1$, so that we get $P_i = V(i,i)$. Note that $P_0 = 0$. Solving the recurrence relation (2) yields

$$\begin{aligned} V(b-1,\delta) &= \left(\frac{d}{d-1}\right)^{\delta-(b-1)} P_{b-1} + \frac{1}{d-1}\left(\frac{1}{b\cdot c} - 1\right)\cdot \sum_{i=0}^{\delta-(b-1)-1}\left(\frac{d}{d-1}\right)^{i} \\ &= \left(\frac{d}{d-1}\right)^{\delta-(b-1)} P_{b-1} + \frac{1}{d-1}\left(\frac{1}{b\cdot c} - 1\right)\cdot \frac{\left(\frac{d}{d-1}\right)^{\delta-(b-1)} - 1}{\left(\frac{d}{d-1}\right) - 1} \\ &= \left(\frac{d}{d-1}\right)^{\delta-(b-1)}\left(P_{b-1} + \frac{1}{b\cdot c} - 1\right) + 1 - \frac{1}{b\cdot c}\,. \end{aligned}$$

The following lemma generalizes the computation above to all other load levels.

▶ **Lemma 1.** *For all l, $0 \leq l \leq b$, and for all δ, $l \leq \delta \leq kb$, it holds that*

$$V(l,\delta) = \sum_{i=l}^{b-1}(-1)^{i-l}\frac{1}{(d-1)^{i-l}}\binom{\delta-l}{i-l}\left(\frac{d}{d-1}\right)^{\delta-i}\left(P_i + \frac{b-i}{b\cdot c} - 1\right) + 1 - \frac{b-l}{b\cdot c}\,. \quad (3)$$

Proof. By induction over l, starting with $l = b$ and going down to $l = 0$. The induction base $l = b$ is true, because we have $V(b,\delta) = 1$. Thus, we focus on the induction step $l+1 \rightsquigarrow l$. Similar arguments as before yield for $l \leq \delta < kb$

$$q(l,\delta) = \frac{1}{d-1}\left(\frac{1}{b\cdot c} - p(l,\delta)\right) = \frac{1}{d-1}\left(\frac{1}{b\cdot c} - V(l+1,\delta+1) + V(l,\delta)\right)\,.$$

We can now define the recurrence relation for $V(l,\delta)$

$$V(l,\delta+1) = V(l,\delta) + q(l,\delta) = \frac{d}{d-1}V(l,\delta) + \frac{1}{d-1}\left(\frac{1}{b\cdot c} - V(l+1,\delta+1)\right)\,,$$

with $V(l,l) = P_l$. Solving this recurrence yields

$$\begin{aligned} V(l,\delta) = &\left(\frac{d}{d-1}\right)^{\delta-l} P_l + \frac{1}{d-1}\frac{1}{b\cdot c}\sum_{i=0}^{\delta-l-1}\left(\frac{d}{d-1}\right)^i \\ &- \frac{1}{d-1}\sum_{i=0}^{\delta-l-1}\left(\frac{d}{d-1}\right)^i V(l+1,\delta-i)\,. \end{aligned} \tag{4}$$

In the next step, we will need the following fact [11].

▶ **Fact 2.** *For $n,k \in \mathbb{N}_0$, it holds that*

$$\sum_{i=0}^{n}\binom{i}{k} = \binom{n+1}{k+1}.$$

Next, we apply the induction hypothesis to determine $V(l,\delta)$. For clarity, we focus on the last sum of equality (4) first

$$\begin{aligned} &\sum_{i=0}^{\delta-l-1}\left(\frac{d}{d-1}\right)^i V(l+1,\delta-i) \overset{\text{IH}}{=} \sum_{i=0}^{\delta-l-1}\left(\frac{d}{d-1}\right)^i \left[\sum_{j=l+1}^{b-1}(-1)^{j-(l+1)}\frac{1}{(d-1)^{j-(l+1)}}\right. \\ &\quad\left.\cdot\binom{\delta-i-(l+1)}{j-(l+1)}\left(\frac{d}{d-1}\right)^{\delta-i-j}\left(P_j+\frac{b-j}{b\cdot c}-1\right)+1-\frac{b-(l+1)}{b\cdot c}\right] \\ &= \sum_{j=l+1}^{b-1}\left[(-1)^{j-(l+1)}\frac{1}{(d-1)^{j-(l+1)}}\left(\frac{d}{d-1}\right)^{\delta-j}\left(P_j+\frac{b-j}{b\cdot c}-1\right)\right. \\ &\quad\left.\cdot\sum_{i=0}^{\delta-l-1}\binom{\delta-i-(l+1)}{j-(l+1)}\right]+\left(1-\frac{b-(l+1)}{b\cdot c}\right)\sum_{i=0}^{\delta-l-1}\left(\frac{d}{d-1}\right)^i \\ &= \sum_{j=l+1}^{b-1}(-1)^{j-(l+1)}\frac{1}{(d-1)^{j-(l+1)}}\left(\frac{d}{d-1}\right)^{\delta-j}\left(P_j+\frac{b-j}{b\cdot c}-1\right)\sum_{i=0}^{\delta-l-1}\binom{i}{j-(l+1)} \\ &\quad+\left(1-\frac{b-(l+1)}{b\cdot c}\right)(d-1)\left(\left(\frac{d}{d-1}\right)^{\delta-l}-1\right) \\ &= \sum_{j=l+1}^{b-1}(-1)^{j-(l+1)}\frac{1}{(d-1)^{j-(l+1)}}\left(\frac{d}{d-1}\right)^{\delta-j}\left(P_j+\frac{b-j}{b\cdot c}-1\right)\binom{\delta-l}{j-l} \\ &\quad+\left(1-\frac{b-(l+1)}{b\cdot c}\right)(d-1)\left(\left(\frac{d}{d-1}\right)^{\delta-l}-1\right), \end{aligned}$$

where we used Fact 2 in the last step. Now, we can finish the induction step by plugging this into (4)

$$
\begin{aligned}
V(l,\delta) &= \left(\frac{d}{d-1}\right)^{\delta-l} P_l + \frac{1}{b\cdot c}\left(\left(\frac{d}{d-1}\right)^{\delta-l} - 1\right) \\
&\quad + \sum_{j=l+1}^{b-1} (-1)^{j-l} \frac{1}{(d-1)^{j-l}} \left(\frac{d}{d-1}\right)^{\delta-j} \left(P_j + \frac{b-j}{b\cdot c} - 1\right) \binom{\delta-l}{j-l} \\
&\quad - \left(1 - \frac{b-(l+1)}{b\cdot c}\right)\left(\left(\frac{d}{d-1}\right)^{\delta-l} - 1\right) \\
&= \left(\frac{d}{d-1}\right)^{\delta-l} \left(P_l + \frac{1}{b\cdot c} + \frac{b-(l+1)}{b\cdot c} - 1\right) + 1 - \frac{1}{b\cdot c} - \frac{b-(l+1)}{b\cdot c} \\
&\quad + \sum_{j=l+1}^{b-1} (-1)^{j-l} \frac{1}{(d-1)^{j-l}} \left(\frac{d}{d-1}\right)^{\delta-j} \left(P_j + \frac{b-j}{b\cdot c} - 1\right) \binom{\delta-l}{j-l} \\
&= \sum_{j=l}^{b-1} (-1)^{j-l} \frac{1}{(d-1)^{j-l}} \left(\frac{d}{d-1}\right)^{\delta-j} \left(P_j + \frac{b-j}{b\cdot c} - 1\right) \binom{\delta-l}{j-l} + 1 - \frac{b-l}{b\cdot c}\,. \qquad \blacktriangleleft
\end{aligned}
$$

So far, we have only leveraged our constraint $V(b,\cdot) = 1$. With our description of $V(l,\delta)$, for all $0 \le l \le b$ and $l \le \delta \le kb$, we can also leverage $V(\cdot, kb) = 1$ to determine P_i, for all $0 \le i \le b-1$. For this, we will need the following technical lemma. The proof is given in the full version of the paper.

▶ **Lemma 3.** *For $k, n, m \in \mathbb{N}$, with $m \ge n \ge k$, it holds that*

$$
\sum_{i=1}^{n-k} (-1)^i \binom{m}{i} \binom{m-i}{n-k-i} = -\binom{m}{n-k}\,.
$$

▶ **Lemma 4.** *For all l, $0 \le l \le b-1$, it holds that*

$$
\left(\frac{d}{d-1}\right)^{kb-l} \left(P_l + \frac{b-l}{b\cdot c} - 1\right) = \frac{1}{b\cdot c}\left(\sum_{i=1}^{b-l} i \binom{kb-l}{b-l-i} \frac{1}{(d-1)^{b-l-i}}\right)\,. \tag{5}
$$

Proof. By induction over l from $l = b-1$ down to $l = 0$. We start with the induction base $l = b-1$. Our second constrain yields $V(b-1, kb) = 1$. It then follows from Lemma 1 that

$$
V(b-1,kb) = \left(\frac{d}{d-1}\right)^{kb-(b-1)} \left(P_{b-1} + \frac{1}{b\cdot c} - 1\right) + 1 - \frac{1}{b\cdot c} \overset{\text{def.}}{=} 1\,,
$$

which immediately finishes the induction base, since the right-hand side of (5) is simply $1/(b\cdot c)$.

We can now move on to the induction step $\forall i > l \rightsquigarrow l$. We use $V(l, kb) = 1$ and rearrange with the help of Lemma 1

$$
\begin{aligned}
&\left(\frac{d}{d-1}\right)^{kb-l} \left(P_l + \frac{b-l}{b\cdot c} - 1\right) = \frac{b-l}{b\cdot c} \\
&\quad - \sum_{i=l+1}^{b-1} (-1)^{i-l} \frac{1}{(d-1)^{i-l}} \binom{kb-l}{i-l} \left(\frac{d}{d-1}\right)^{kb-i} \left(P_i + \frac{b-i}{b\cdot c} - 1\right)\,.
\end{aligned} \tag{6}
$$

It is now possible to apply the induction hypothesis. For clarity, we focus on the second line of (6)

$$\sum_{i=l+1}^{b-1}(-1)^{i-l}\frac{1}{(d-1)^{i-l}}\binom{kb-l}{i-l}\frac{1}{b\cdot c}\left(\sum_{j=1}^{b-i}j\binom{kb-i}{b-i-j}\frac{1}{(d-1)^{b-i-j}}\right)$$
$$=\frac{1}{b\cdot c}\left(\sum_{a=1}^{b-l-1}(-1)^a\sum_{j=1}^{b-(l+a)}j\frac{1}{(d-1)^{b-l-j}}\binom{kb-l}{a}\binom{kb-(l+a)}{b-(l+a)-j}\right),$$

where we substituted $a := i-l$. Next, we carefully swap these nested sums. For this, observe that, for a fixed value j, we have exactly $b-l-j$ addends, more precisely, one addend for each $1 \le a \le b-l-j$

$$\frac{1}{b\cdot c}\left(\sum_{a=1}^{b-l-1}(-1)^a\sum_{j=1}^{b-(l+a)}j\frac{1}{(d-1)^{b-l-j}}\binom{kb-l}{a}\binom{kb-l-a}{b-l-a-j}\right)$$
$$=\frac{1}{b\cdot c}\left(\sum_{j=1}^{b-l-1}j\frac{1}{(d-1)^{b-l-j}}\sum_{a=1}^{b-l-j}(-1)^a\binom{kb-l}{a}\binom{kb-l-a}{b-l-a-j}\right)$$
$$=-\frac{1}{b\cdot c}\left(\sum_{j=1}^{b-l-1}j\frac{1}{(d-1)^{b-l-j}}\binom{kb-l}{b-l-j}\right),$$

where we applied Lemma 3 with $k=j$, $n=b-l$ and $m=kb-l$ in the last step. Plugging this back into (6) gives

$$\left(\frac{d}{d-1}\right)^{kb-l}\left(P_l+\frac{b-l}{b\cdot c}-1\right)=\frac{b-l}{b\cdot c}+\frac{1}{b\cdot c}\left(\sum_{j=1}^{b-l-1}j\frac{1}{(d-1)^{b-l-j}}\binom{kb-l}{b-l-j}\right)$$
$$=\frac{1}{b\cdot c}\left(\sum_{j=1}^{b-l}j\frac{1}{(d-1)^{b-l-j}}\binom{kb-l}{b-l-j}\right). \qquad \blacktriangleleft$$

With the help of Lemma 4, we can finally determine the resulting constant c^*. For $l=0$, we have $P_0=0$, and thus

$$\left(\frac{d}{d-1}\right)^{kb}\left(\frac{1}{c^*}-1\right)=\frac{1}{b\cdot c^*}\left(\sum_{i=1}^{b}i\binom{kb}{b-i}\frac{1}{(d-1)^{b-i}}\right),$$

where solving for c^* yields

$$c^*=1-\frac{1}{b}\left(\sum_{i=1}^{b}i\binom{kb}{b-i}\frac{1}{(d-1)^{b-i}}\right)\left(1-\frac{1}{d}\right)^{kb}.$$

▶ **Theorem 5.** *WeightedAssignment achieves a competitive ratio of c^* for the online b-matching problem with uniform server capacities on (k,d)-graphs.*

Lemma 1, together with Lemma 4 and c^*, specifies the function V. Its values can be calculated in a preprocessing step and accessed by table lookup when WeightedAssignment serves requests. The parameters k and d must be known. In an application, they can be learned over time. Alternatively, one can work with conservative estimates. Figure 1 shows

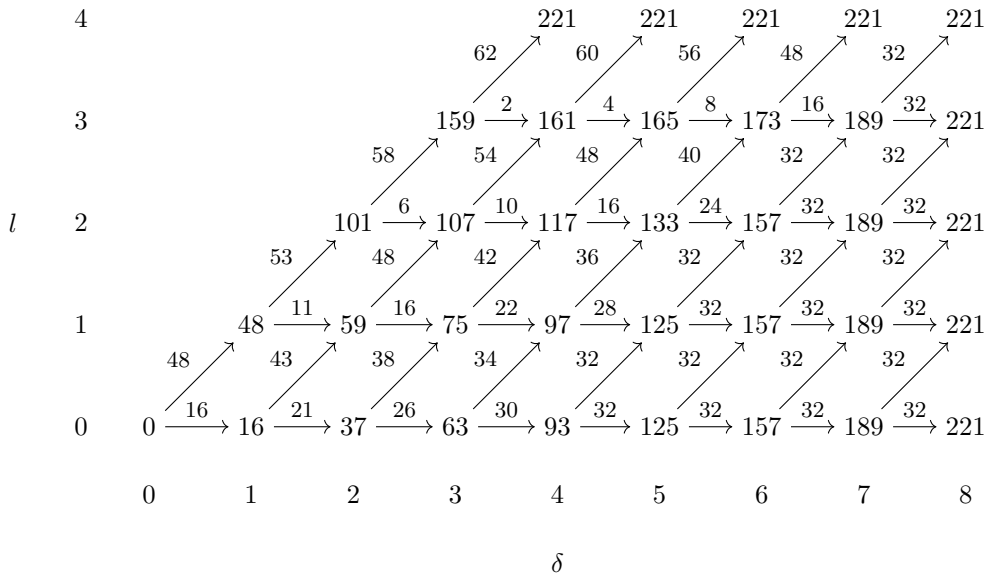

Figure 1 The function V for $k = d = 2$ and $b = 4$. All values are multiplied by 221.

the function V for a small example with $k = d = 2$ and $b = 4$. In this case, we have $c^* = 221/256$ and $1/(bc^*) = 64/221$. The arrows depict the possible increases in V for every l, $0 \leq l < b$, and δ, $l \leq \delta < kb$, i.e. horizontal arrows denote the $q(l, \delta)$ values and diagonal arrows the $p(l, \delta)$ values. All actual values are multiplied by 221 in order to eliminate fractions. Observe that $p(l, \delta) + (d - 1)q(l, \delta) = 1/(bc^*)$ holds for all l and δ. This is what allows us to bound the total increase in the dual solution by $1/c^*$, if we always pick the neighboring server s that maximizes $q(l_s, \delta_s)$. For the matching decisions, WeightedAssignment only uses the horizontal arrows. Notice that WeightedAssignment is different from a Balance algorithm that breaks ties by HighDegree, or from a HighDegree algorithm that breaks ties by Balance. For example, WeightedAssignment prefers a server with load 1 and degree 5 to a server with load 0 and degree 1, whom it then prefers to a server with load 3 and degree 6.

3 Upper bounds

We will show that WeightedAssignment is optimal for (k, d)-graphs with $k \geq d$, i.e. no deterministic online algorithm can achieve a competitive ratio better than c^*. We start by proving this for the online b-matching problem with uniform server capacities, and later extend it to the more general problems.

First, we show that any (k, d)-graph with uniform server capacities b has a perfect b-matching, i.e. a matching where every server $s \in S$ is matched exactly b times, if $k \geq d$. This generalizes Lemma 6.1 in [19], which states this for $b = 1$. The proof is given in the full version of this paper.

▶ **Lemma 6.** *Every (k, d)-graph $G = (S \cup R, E)$, where $k \geq d$, with uniform server capacities b has a perfect b-matching.*

We move on to describing the adversary input. We start by following the construction of the previously known upper bound detailed in [19]. There are $N = d^{kb}$ servers, and the requests arrive in kb rounds. Let S_i denote the set of unmatched servers at the beginning

of round i, $0 \leq i < kb$. It will hold that every server in S_i has a current degree of i and that $|S_i| = N(1-1/d)^i$. Note that this number is always a multiple of d, by choice of N. During round i, $|S_i|/d$ requests are introduced, such that every request is adjacent to exactly d distinct servers in S_i and every server in S_i gains one new neighboring request. If the online algorithm decides not to match an introduced request, we consider it matched to an arbitrary neighbor. This will only improve the performance of said algorithm. This means that a $(1-1/d)$ fraction of the servers in S_i are still unmatched after round i, explaining the previously mentioned $|S_i| = N(1-1/d)^i$. Thus, there are $N \cdot (1-1/d)^{kb}$ servers with degree kb and load 0 after round $kb-1$, irrespective of what choices the algorithm makes. In a final round, further requests are introduced to all matched servers arbitrarily, such that we get a valid (k,d)-graph. The online algorithm will have matched at most $bN \cdot \left(1-(1-1/d)^{kb}\right)$ requests, while Lemma 6 implies that an optimal offline algorithm matches bN requests. This yields the previously known upper bound of $\left(1-(1-1/d)^{kb}\right)$.

However, it seems suboptimal to introduce requests arbitrarily after the initial kb rounds. In fact, we will show that we can further limit the number of requests matched by the online algorithm if we introduce the requests more carefully. Note that all the servers that are matched during round i of the previous input are similar in the sense that they all have degree i and load 1. We will apply the ideas above recursively to the sets of matched servers of all rounds. More precisely, let T denote the set of matched server during some round. Say all servers in T have load l, $0 \leq l < b$, and degree δ, $l \leq \delta < kb$. We then schedule $kb-\delta$ rounds for T using the same construction as above. Let T_j denote the set of servers that still have load l at the beginning of round j, $0 \leq j < kb-\delta$. It will now hold that every server in T_j has a current degree of $\delta + j$ and that $|T_i| = |T|(1-1/d)^i$. We have to make sure that all the possible values of $|T_i|$ are multiples of d. This is done by increasing the initial number of servers N adequately. After these $kb-\delta$ round, we have $|T| \cdot (1-1/d)^{kb-\delta}$ servers with degree kb and load l. This process is repeated for all the sets of matched servers until we eventually obtain a valid (k,d)-graph, in which every server has degree kb.

For this, we formally define a function F, where $F(x,l,\delta)$ denotes how many units of capacity we can force a deterministic online algorithm to leave empty when starting with x servers that all have load l and degree δ. This allows us to upper bound the number of matched requests by $bN - F(N,0,0)$, yielding the following upper bound

$$c \leq \frac{bN - F(N,0,0)}{bN} = 1 - \frac{F(N,0,0)}{bN}\,. \tag{7}$$

We cannot create any empty spots on full servers, so we have $F(x,b,\delta) = 0$, for all $b \leq \delta \leq kb$. Once a server has kb adjacent requests, we have satisfied the (k,d)-graph property locally, so we do not need to introduce any more adjacent requests for this server. This is captured by $F(x,l,kb) = x \cdot (b-l)$, for all $0 \leq l \leq b$. For all other combinations of l and δ, it is possible to define F recursively. Recall that we introduce $kb-\delta$ rounds when starting with x servers all with load l and degree δ. During each of these rounds, exactly a $1/d$ fraction of the servers that still had load l at the start of the round are discarded. Moreover, every server gets exactly one new neighbor during each round until they are matched. This implies

$$F(x,l,\delta) = x \cdot \left(1-\frac{1}{d}\right)^{kb-\delta}(b-l) + \sum_{i=1}^{kb-\delta} F\left(x \cdot \frac{1}{d}\left(1-\frac{1}{d}\right)^{i-1}, l+1, \delta+i\right)\,, \tag{8}$$

for all $0 \leq l < b$ and $l \leq \delta < kb$. The following lemma solves this recurrence, which we can then apply in (7) to obtain the theorem.

▶ **Lemma 7.** *For all l, $0 \le l \le b$, and all δ, $l \le \delta \le kb$, it holds that*

$$F(x,l,\delta) = x\left(1-\frac{1}{d}\right)^{kb-\delta}\left(\sum_{i=1}^{b-l} i\binom{kb-\delta}{b-l-i}\frac{1}{(d-1)^{b-l-i}}\right). \tag{9}$$

Proof. By induction over l, starting with $l = b$ and going down to $l = 0$. The induction base is satisfied as we get the empty sum in (9), making the whole expression zero. In the induction step $l+1 \rightsquigarrow l$, we can apply our induction hypothesis to F in (8).

$$\begin{aligned}
&\sum_{i=1}^{kb-\delta} F\left(x\cdot\frac{1}{d}\left(1-\frac{1}{d}\right)^{i-1}, l+1, \delta+i\right)\\
&\overset{\text{IH}}{=} \sum_{i=1}^{kb-\delta} x\frac{1}{d}\left(1-\frac{1}{d}\right)^{i-1}\left(1-\frac{1}{d}\right)^{kb-(\delta+i)}\left(\sum_{j=1}^{b-(l+1)} j\binom{kb-\delta-i}{b-(l+1)-j}\frac{1}{(d-1)^{b-(l+1)-j}}\right)\\
&= x\frac{1}{d}\left(1-\frac{1}{d}\right)^{kb-\delta-1}\left(\sum_{j=1}^{b-(l+1)} j\frac{1}{(d-1)^{b-(l+1)-j}}\sum_{i=1}^{kb-\delta}\binom{kb-\delta-i}{b-(l+1)-j}\right)\\
&= x\left(1-\frac{1}{d}\right)^{kb-\delta}\frac{1}{d-1}\left(\sum_{j=1}^{b-(l+1)} j\frac{1}{(d-1)^{b-l-j-1}}\sum_{i=0}^{kb-\delta-1}\binom{i}{b-l-j-1}\right)\\
&= x\left(1-\frac{1}{d}\right)^{kb-\delta}\left(\sum_{j=1}^{b-(l+1)} j\frac{1}{(d-1)^{b-l-j}}\binom{kb-\delta}{b-l-j}\right),
\end{aligned}$$

where we used Fact 2 in the last step. Finally, we can plug this result back into (8) to obtain

$$\begin{aligned}
F(x,l,\delta) &= x\cdot\left(1-\frac{1}{d}\right)^{kb-\delta}(b-l) + \sum_{i=1}^{kb-\delta} F\left(x\cdot\frac{1}{d}\left(1-\frac{1}{d}\right)^{i-1}, l+1, \delta+i\right)\\
&= x\cdot\left(1-\frac{1}{d}\right)^{kb-\delta}\left((b-l) + \sum_{j=1}^{b-(l+1)} j\frac{1}{(d-1)^{b-l-j}}\binom{kb-\delta}{b-l-j}\right)\\
&= x\cdot\left(1-\frac{1}{d}\right)^{kb-\delta}\left(\sum_{j=1}^{b-l} j\frac{1}{(d-1)^{b-l-j}}\binom{kb-\delta}{b-l-j}\right).
\end{aligned}$$

◀

▶ **Theorem 8.** *No deterministic online algorithm for the b-matching problem with uniform server capacities b can achieve a competitiveness better than c^* on (k,d)-graphs with $k \ge d$.*

We extend this upper bound to the more general case with variable server capacities, which we will examine in the next section. Let $b_{min} = \min_{s\in S} b_s$. The optimal competitiveness is then

$$c^*_{\min} = 1 - \frac{1}{b_{\min}}\left(\sum_{i=1}^{b_{\min}} i\binom{kb_{\min}}{b_{\min}-i}\frac{1}{(d-1)^{b_{\min}-i}}\right)\left(1-\frac{1}{d}\right)^{kb_{\min}}.$$

▶ **Corollary 9.** *No deterministic online algorithm for the b-matching problem with variable server capacities can achieve a competitive ratio better than $c^*_{\min}$ on (k,d)-graphs with $k \ge d$.*

The proof is detailed in the full version of this paper. Obviously, Theorem 8 and Corollary 9 also hold for the more general vertex-weighted b-matching problem, addressed in the next section.

4 Variable server capacities and vertex weights

We detail the necessary changes to WeightedAssignment such that it can handle variable server capacities as well as vertex weights, while still achieving the optimal competitive ratio.

4.1 Variable server capacities

Recall that every server $s \in S$ now has a server capacity b_s, and thus a degree of at least $d(s) \geq k \cdot b_s$. This changes the objective function of the dual linear program to

$$\sum_{s \in S} b_s \cdot x(s) + \sum_{r \in R} y(r) \,.$$

We handle this by computing the function V_s for every server individually, for its capacity b_s. This means that we construct V_s according to Section 2, such that

$$b_s \cdot \Big(V_s(l+1, \delta+1) - V_s(l, \delta) + (d-1) \cdot \big(V_s(l, \delta+1) - V_s(l, \delta) \big) \Big) \leq \frac{1}{c_s^*} \,, \tag{10}$$

where c_s^* is equal to c^*, but b is replaced by b_s. Moreover, we have $V_s(b_s, \cdot) = 1$ and $V_s(\cdot, \delta') = 1$ if $\delta' \geq k b_s$, meaning that we again construct a feasible dual solution. However, we still have to adapt the decision criterion of WeightedAssignment. We change Line 7 in Algorithm 1 to

$$\text{Match } r \text{ to } \arg\max \big\{ b_s \cdot \big(V_s(l_s, \delta_s + 1) - V_s(l_s, \delta_s) \big) : s \in N(r) \big\} \,.$$

This allows us to upper bound the total increase in the dual solution when the adapted strategy, called WeightedAssignment(VC), assigns a request r to a server s by

$$\begin{aligned} \Delta D &= b_s \cdot \Big(V_s(l_s + 1, \delta_s + 1) - V_s(l_s, \delta_s) \Big) \\ &\quad + \sum_{s' \in N(r) \setminus \{s\}} b_{s'} \cdot \Big(V_{s'}(l_{s'}, \delta_{s'} + 1) - V_{s'}(l_{s'}, \delta_{s'}) \Big) \\ &\leq b_s \cdot \Big(V_s(l_s + 1, \delta_s + 1) - V_s(l_s, \delta_s) + (d-1) \cdot \big(V_s(l_s, \delta_s + 1) - V_s(l_s, \delta_s) \big) \Big) \leq \frac{1}{c_s^*} \,. \end{aligned}$$

Thus, WeightedAssignment(VC) achieves a competitive ratio of $\min_{s \in S} c_s^*$. In Section 5 we will show that c^* is monotonically increasing in b for $k \geq d$, meaning that $\min_{s \in S} c_s^* = c_{\min}^*$, cf. Section 3.

▶ **Theorem 10.** *WeightedAssignment(VC) achieves a competitive ratio of* $\min_{s \in S} c_s^*$ *the b-matching problem with variable server capacities on* (k, d)*-graphs. The ratio equals* $c_{\min}^*$ *and is optimal for* $k \geq d$.

4.2 Vertex weights

At last, we consider the vertex-weighted extension of the online b-matching problem. Every server $s \in S$ now has a weight w_s assigned to it, and the value of every matching edge incident to s is multiplied by w_s. This problem is modelled by the following linear programs.

$$\textbf{Primal:}\quad \max \sum_{\{s,r\}\in E} w_s \cdot m(s,r)$$
$$\text{s.t.} \sum_{r:\{s,r\}\in E} w_s \cdot m(s,r) \le w_s \cdot b_s, \qquad (\forall s \in S)$$
$$\sum_{s:\{s,r\}\in E} m(s,r) \le 1, \qquad (\forall r \in R)$$
$$m(s,r) \ge 0, \qquad (\forall \{s,r\} \in E)\,.$$

$$\textbf{Dual:}\quad \min \sum_{s\in S} w_s \cdot b_s \cdot x(s) + \sum_{r\in R} y(r)$$
$$\text{s.t. } w_s \cdot x(s) + y(r) \ge w_s, \qquad (\forall \{s,r\} \in E)$$
$$x(s), y(r) \ge 0, \qquad (\forall s \in S, \forall r \in R)\,.$$

Observe that $x(s) = 1$, for all $s \in S$, by the end of the algorithm still implies dual feasibility. Hence, we do not need to change the construction of V_s, and still obtain a feasible dual solution. All we need to do is to change the decision criterion of WeightedAssignment(VC) once more to

$$\text{Match } r \text{ to } \arg\max \left\{ w_s \cdot b_s \cdot \big(V_s(l_s, \delta_s + 1) - V_s(l_s, \delta_s)\big) : s \in N(r) \right\}.$$

Whenever the resulting algorithm WeightedAssignment(VW) assigns a request r to a server s, we increase the primal solution by w_s, while we can upper bound the increase in the dual solution by

$$\Delta D \le w_s b_s \cdot \Big(V_s(l_s + 1, \delta_s + 1) - V_s(l_s, \delta_s) + (d-1)\big(V_s(l_s, \delta_s + 1) - V_s(l_s, \delta_s)\big)\Big) \le \frac{w_s}{c^*_s}\,.$$

This again yields a competitiveness of $\min_{s\in S} c^*_s$.

▶ **Theorem 11.** *WeightedAssignment(VW) achieves a competitive ratio of* $\min_{s\in S} c^*_s$ *for the vertex-weighted b-matching problem with variable server capacities on* (k,d)*-graphs. The ratio equals* $c^*_{\min}$ *and is optimal for* $k \ge d$*.*

5 Analysis of the competitive ratio

▶ **Theorem 12.** *If* $k \ge d \ge 2$*, the competitive ratio* c^* *converges to one as* b *tends to infinity, that is* $\lim_{b\to\infty} c^* = 1$*.*

▶ **Theorem 13.** *If* $k \ge d \ge 2$*, the competitive ratio* c^* *is strictly increasing in* b*, for* $b \ge 1$*.*

The proofs of these two theorems are given in the full version of the paper. Theorem 12 is shown with the help of Gaussian hypergeometric functions. They allow us to upper bound c^* with a closed expression. The convergence of this expression can then be shown with the help of Stirling's approximation. Since this does not prove monotonicity, we consider the fraction of c^* with $b+1$ over c^* with b in the proof of Theorem 13. We show that this fraction is lower bounded by 1, again with the help of hypergeometric functions.

References

1 G. Aggarwal, G. Goel, C. Karande, and A. Mehta. Online vertex-weighted bipartite matching and single-bid budgeted allocations. In *Proc. 22nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1253–1264. SIAM, 2011.

2 Y. Azar, I.R. Cohen, and A. Roytman. Online lower bounds via duality. In *Proc. 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1038–1050. SIAM, 2017.

3 N. Buchbinder, K. Jain, and J. Naor. Online primal-dual algorithms for maximizing ad-auctions revenue. In *Proc. 15th Annual European Symposium on Algorithms (ESA)*, Springer LNCS, Volume 4698, pages 253–264, 2007.

4 K. Chaudhuri, C. Daskalakis, R.D. Kleinberg, and H. Lin. Online bipartite perfect matching with augmentations. In *Proc. 28th IEEE International Conference on Computer Communications (INFOCOM)*, pages 1044–1052, 2009.

5 I.R. Cohen and D. Wajc. Randomized online matching in regular graphs. In *Proc. 29th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 960–979. SIAM, 2018.

6 R. Cole, K. Ost, and S. Schirra. Edge-coloring bipartite multigraphs in $O(E \log D)$ time. *Comb.*, 21(1):5–12, 2001.

7 J. Csima and L. Lovász. A matching algorithm for regular bipartite graphs. *Discret. Appl. Math.*, 35(3):197–203, 1992.

8 N.R. Devanur and T.P. Hayes. The adwords problem: online keyword matching with budgeted bidders under random permutations. In *Proc. 10th ACM Conference on Electronic Commerce (EC)*, pages 71–78. ACM, 2009.

9 N.R. Devanur, B. Sivan, and Y. Azar. Asymptotically optimal algorithm for stochastic adwords. In *Proc. 13th ACM Conference on Electronic Commerce (EC)*, pages 388–404. ACM, 2012.

10 A. Goel, M. Kapralov, and S. Khanna. Perfect matchings in $O(n \log n)$ time in regular bipartite graphs. *SIAM J. Comput.*, 42(3):1392–1404, 2013.

11 R.L. Graham, D.E. Knuth, and O. Patashnik. *Concrete Mathematics: A Foundation for Computer Science*. Addison-Wesley, 1989.

12 E.F. Grove, M.-Y. Kao, P. Krishnan, and J.S. Vitter. Online perfect matching and mobile computing. In *Proc. 4th International Workshop on Algorithms and Data Structures (WADS)*, LNCS, Volume 955, pages 194–205. Springer, 1995.

13 P. Jaillet and X. Lu. Online stochastic matching: New algorithms with better bounds. *Math. Oper. Res.*, 39(3):624–646, 2014.

14 B. Kalyanasundaram and K. Pruhs. An optimal deterministic algorithm for online b-matching. *Theor. Comput. Sci.*, 233(1-2):319–325, 2000.

15 R.M. Karp, U.V. Vazirani, and V.V. Vazirani. An optimal algorithm for on-line bipartite matching. In *Proc. 22nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 352–358, 1990.

16 M. Mahdian and Q. Yan. Online bipartite matching with random arrivals: An approach based on strongly factor-revealing LPs. In *Proc. 43rd ACM Symposium on Theory of Computing (STOC)*, pages 597–606, 2011.

17 V.H. Manshadi, S. Oveis Gharan, and A. Saberi. Online stochastic matching: Online actions based on offline statistics. *Math. Oper. Res.*, 37(4):559–573, 2012.

18 A. Mehta, A. Saberi, U.V. Vazirani, and V.V. Vazirani. Adwords and generalized online matching. *J. ACM*, 54(5):22, 2007.

19 J. Naor and D. Wajc. Near-optimum online ad allocation for targeted advertising. *ACM Trans. Economics and Comput.*, 6(3-4):16:1–16:20, 2018.

20 A. Schrijver. Bipartite edge coloring in $O(\Delta m)$ time. *SIAM J. Comput.*, 28(3):841–846, 1998.

21 V.V. Vazirani. Randomized online algorithms for Adwords. *CoRR*, abs/2107.10777, 2021. `arXiv:2107.10777`.