Optimal Algorithms for Online b-Matching with Variable 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, which generalizes classical online matching introduced by Karp, Vazirani and Vazirani (STOC 1990). Consider a bipartite graph $G = (S \dot\cup R, E)$. Every vertex $s \in S$ is a server with a capacity b_s, indicating the number of possible matching partners. The vertices $r \in R$ are requests that arrive online and must be matched immediately to an eligible server. The goal is to maximize the cardinality of the constructed matching. In contrast to earlier work, we study the general setting where servers may have arbitrary, individual capacities. We prove that the most natural and simple online algorithms achieve optimal competitive ratios.

As for deterministic algorithms, we give a greedy algorithm RelativeBalance and analyze it by extending the primal-dual framework of Devanur, Jain and Kleinberg (SODA 2013). In the area of randomized algorithms we study the celebrated Ranking algorithm by Karp, Vazirani and Vazirani. We prove that the original Ranking strategy, simply picking a random permutation of the servers, achieves an optimal competitiveness of $1 - 1/e$, independently of the server capacities. Hence it is not necessary to resort to a reduction, replacing every server s by b_s vertices of unit capacity and to then run Ranking on this graph with $\sum_{s \in S} b_s$ vertices on the left-hand side. From a theoretical point of view our result explores the power of randomization and strictly limits the amount of required randomness. From a practical point of view it leads to more efficient allocation algorithms.

Technically, we show that the primal-dual framework of Devanur, Jain and Kleinberg cannot establish a competitiveness better than 1/2 for the original Ranking algorithm, choosing a permutation of the servers. Therefore, we formulate a new configuration LP for the b-matching problem and then conduct a primal-dual analysis. We extend this analysis approach to the vertex-weighted b-matching problem. Specifically, we show that the algorithm PerturbedGreedy by Aggarwal, Goel, Karande and Mehta (SODA 2011), again with a sole randomization over the set of servers, is $(1 - 1/e)$-competitive. Together with recent work by Huang and Zhang (STOC 2020), our results demonstrate that configuration LPs can be strictly stronger than standard LPs in the analysis of more complex matching problems.

2012 ACM Subject Classification Theory of computation → Online algorithms

Keywords and phrases Online algorithms, primal-dual analysis, configuration LP, b-matching, variable vertex capacities, unweighted matching, vertex-weighted matching

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2021.2

Category APPROX

Funding Work supported by the European Research Council, Grant Agreement No. 691672.

[1] Corresponding author

1 Introduction

Matching in bipartite graphs is a fundamental problem with numerous applications in computer science. We study the *b-matching problem* [13], where the vertices of one set of the bipartition may be matched multiple times. It generalizes the standard matching problem. Furthermore, it models capacitated allocations as well as interesting special cases of the timely AdWords problem.

More specifically, let $G = (S \dot{\cup} R, E)$ be a bipartite graph. The vertices of S are servers. Each server $s \in S$ has an individual capacity b_s, indicating the maximum number of possible matching partners. The vertices of R are requests that have to be assigned to the servers. We consider the online problem where the set S of servers is known in advance and the requests of R arrive sequentially 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.

Prior work on b-matchings has mostly focused on the case that all servers have the same capacity, i.e. $b_s = b$, for all $s \in S$. In this paper we study the general setting of individual server capacities, as described above. This setting is particularly relevant in applications. Furthermore, we 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. The goal is to maximize the total weight of the constructed matching. Again this extension is interesting in allocation problems.

If $b_s = 1$ for all $s \in S$, the b-matching problem is equal to classic online bipartite matching, which was introduced in a seminal paper by Karp et al. [15] and has received tremendous research interest over the last 30 years. The b-matching problem models a range of interesting applications. Naturally, the servers can be compute servers that process persistent jobs arriving over time. Furthermore, the servers can be facilities that stream content online, host web pages or store data remotely [5]. More generally, the servers can represent stations in mobile computing, queues in a network switch or even locations in a hash table [2, 5, 9]. Obviously, each server can only handle a limited number of clients.

Another relevant application are the AdWords problem and ad auctions in search engine companies [18]. There is a set of advertisers, each with a daily budget, who wish to link their ads to search keywords and issue respective bids. Queries along with their keywords arrive online and must be allocated instantly to the advertisers. The b-matching problem models the basic setting where all bids 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.

We analyze the performance of algorithms for the b-matching problem 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. If Alg is a randomized algorithm, then $\textsc{Alg}(G)$ has to be replaced by the expected value $\mathbb{E}[\textsc{Alg}(G)]$.

Related Work. Straightforward arguments show that any algorithm that matches an incoming request to an eligible server with remaining capacity, if there exists one, is $\frac{1}{2}$-competitive. Kalyanasundaram and Pruhs [13] investigate the b-matching problem if all servers have equal capacity, i.e. $b_s = b$ for all $s \in S$. They present a deterministic Balance algorithm that matches a new request to an adjacent server whose current load is smallest. Kalyanasundaram and Pruhs prove that Balance achieves an optimal competitive ratio

of $1 - 1/(1 + 1/b)^b$. As b grows, the latter expression tends to $1 - 1/e \approx 0.63$. Azar and Litichevsky [2] give an alternative analysis of the BALANCE algorithm. Chaudhuri et al. [5] and Grove et al. [9] study 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 where a request is reassigned to a different server.

In a famous paper, Karp et al. [15] introduced the online bipartite matching problem. This is a b-matching problem where all servers have a capacity of 1, i.e. each vertex in the graph may be incident to at most one matching edge. Online bipartite matching has received tremendous research interest over the last years and we only mention the most important results relevant to our work. Again, any algorithm that matches an incoming request to an arbitrary available partner is $\frac{1}{2}$-competitive. No deterministic online algorithm can be better than $\frac{1}{2}$-competitive. Karp et al. [15] show that an algorithm RANDOM, which matches a request to an available partner chosen uniformly at random, does not achieve a competitiveness greater than $1/2$. As a main result they propose the celebrated RANKING algorithm. This strategy initially chooses a random permutation of the vertices in S. Thereby, each such vertex is assigned a priority or *rank*. Whenever a vertex of R arrives, it is matched to the eligible vertex of highest rank in S. Karp et al. prove that RANKING is $(1 - 1/e)$-competitive. This ratio is best possible for randomized algorithms [15].

Simplified and alternative analyses of RANKING were provided in [1, 3, 6, 7]. In particular, Devanur et al. [6] developed an elegant primal-dual analysis. Aggarwal et al. [1] defined online vertex-weighted bipartite matching, where each vertex $s \in S$ has a weight w_s. Again, all vertices of S have a capacity of 1. The goal is to maximize the total weight of the constructed matching. Aggarwal et al. [1] devise a generalization of RANKING, named PERTURBED-GREEDY, and prove that it is $(1 - 1/e)$-competitive. Devanur et al. [6] analyze this strategy in their compact primal-dual framework. Further work on online bipartite matching considers different input models [8, 12, 14, 16] or refined matching models [10, 17].

The AdWords problem was formally defined by Mehta et al. [18]. They present a deterministic online algorithm that achieves a competitive ratio of $1 - 1/e$, under the assumption that the bids are small compared to the advertisers' budgets. No randomized algorithm can obtain a better competitive factor. Buchbinder et al. [4] develop a primal-dual algorithm that attains a competitiveness of $(1 - 1/c)(1 - R_{\max})$, where $c = (1 + R_{\max})^{1/R_{\max}}$ and $R_{\max}$ is the maximum ratio between the bid of any advertiser and its total budget. Huang et al. [11] give a 0.5016-competitive algorithm, for AdWords without the small-bids assumption.

Our Contributions. We present a comprehensive study of the b-matching problem with variable server capacities. As a main contribution we show that the most natural and simple online algorithms obtain optimal competitive ratios.

First, we concentrate on the unweighted setting, with the objective to maximize the cardinality of the constructed matching. In Section 2 we study deterministic algorithms. We formulate and analyze a strategy RELATIVEBALANCE that assigns an incoming request to an eligible server with minimum *relative load*. The relative load of a server s is the number requests that are currently matched with s divided by the capacity b_s. Thus the algorithm considers which fraction of a server's capacity is already used. This is the most straightforward greedy policy for the setting with variable server capacities. We show that RELATIVEBALANCE achieves a competitive ratio of $1 - 1/(1 + 1/b_{\min})^{b_{\min}}$, where $b_{\min} = \min_s b_s$ is the minimum server capacity. The performance ratio is best possible for deterministic online algorithms.

In order to evaluate RelativeBalance we conduct a primal-dual analysis. We extend the framework by Devanur et al. [6], this time to analyze a deterministic algorithm different from Ranking. We remark that Balance by Kalyanasundaram and Pruhs [13] does not achieve a competitive ratio of $1-1/(1+1/b_{\min})^{b_{\min}}$ when using only $b_{\min}$ spots of each server because Opt may use the additional capacity. Moreover, we would like to add that the results by Buchbinder et al. [4] also imply a deterministic online algorithm with a competitiveness of $1-1/(1+1/b_{\min})^{b_{\min}}$ for the b-matching problem. However, their algorithm is not equal to RelativeBalance. In fact, their strategy may assign a request to a server not having the smallest relative load and does not necessarily use the full capacity of a server, leaving requests unmatched. This leads to somewhat unintuitive assignments. We give details in Appendix A. Of course, Buchbinder et al. [4] were interested in the general AdWords problem and did not tailor their analysis to b-matchings.

In Section 3 we study randomized online algorithms. In a first step we examine the Random algorithm, which assigns an incoming request to a random adjacent server with remaining capacity. We prove that the competitive factor of Random is not better than 1/2. The major part of Section 3 investigates the original Ranking algorithm. More specifically, Ranking initially picks a random permutation of the servers. An incoming request is matched to the eligible server of highest rank. We prove that Ranking achieves a competitive ratio of $1-1/e$, independently of the server capacities. The ratio of $1-1/e$ is best possible for randomized algorithms [18]. Surprisingly, the original Ranking algorithm has an optimal competitiveness for the more complex b-matching problem. We are not aware of any other generalization of the classical online matching problem where this holds true.

Observe that we can also obtain a competitive ratio of $1-1/e$ using the following reduction to standard online bipartite matching: Replace each server s with capacity b_s by exactly b_s individual vertices of capacity 1. Each request adjacent to s gets incident edges to each of these b_s vertices. On the resulting graph with $\sum_{s\in S} b_s$ vertices on the left-hand side of the bipartition, execute the Ranking algorithm. Such a reduction can also be applied for deterministic online algorithms but only gives a competitive factor of 1/2.

Our result for the original Ranking algorithm, executed on the initial input graph G, has the following implications. (1) From a theoretical point of view, an interesting question is how much randomness is needed to obtain a competitiveness of $1-1/e$. Our analysis demonstrates that a straightforward execution of the barely random Ranking strategy attains this ratio. No randomization over the server spots is necessary. (2) In practical applications a ranking of the servers leads to simple and efficient allocation algorithms. With a random permutation of a huge number of server spots, assignments might be difficult, perhaps even impossible to compute.

In our analysis we first demonstrate that the framework by Devanur et al. [6] cannot establish a competitiveness of $1-1/e$ for Ranking, when executed on the original graph G. It only yields a competitiveness of 1/2. Therefore, as a main technical contribution, we formulate a new configuration linear program (LP) for the b-matching problem. Using this configuration LP, we then conduct a primal-dual analysis by extending the framework of Devanur et al. [6]. We point out that, for the bipartite matchings with stochastic rewards, Huang and Zhang [10] recently were the first to employ configuration LPs. However, the concrete LPs used in [10] and in this paper are different, apart from a modeling of vertex neighborhoods. Also, the analyses differ so as to obtain the desired performance ratios.

In Section 4 we investigate vertex-weighted b-matching, with the objective to maximize the total weight of the constructed matching. We focus on randomized strategies and study Perturbed-Greedy [1], which was introduced for vertex-weighted online bipartite matching,

where each vertex $s \in S$ has a capacity of 1. The algorithm, for each $s \in S$, computes a rank based on an initial random choice. A request is matched to the eligible vertex $s \in S$ of highest rank. We investigate Perturbed-Greedy for the b-matching problem when executed on the original input graph G, without the above reduction of splitting a server s into b_s vertices of unit capacity. We extend our analysis approach based on configuration LPs and prove that the algorithm achieves an optimal competitive ratio of $1 - 1/e$.

In summary, simple rank-based algorithms that make initial random choices for the servers (but not for the server spots) achieve an optimal competitive ratio of $1 - 1/e$, independently of the server capacities. Furthermore, the paper by Huang and Zhang [10] and our work show that configuration LPs can be more powerful than standard LPs in the analysis of more advanced matching problems.

2 Deterministic algorithms for maximum-cardinality b-matching

It is easy to verify that an online algorithm that matches a new request to an eligible server with largest remaining capacity does not achieve a competitiveness greater than 1/2.

In the following we present our natural RelativeBalance algorithm. Let $load_s$ denote the (absolute) load of a server $s \in S$, i.e. the number of requests assigned to s so far. We define the relative server load as $l_s := load_s/b_s$. RelativeBalance simply assigns incoming requests to an eligible neighbor with minimum relative server load.

Algorithm 1 RelativeBalance.

while *a new request* $r \in R$ *arrives* **do**
  Let $N(r)$ denote the set of neighbors of r with remaining capacity;
  if $N(r) = \emptyset$ **then**
    Do not match r;
  else
    Match r to $\arg\min\{l_s : s \in N(r)\}$ (break ties arbitrarily);
  end
end

We analyze RelativeBalance by conducting a primal-dual analysis. For this, consider the classical (relaxed) primal and dual LP of maximum cardinality online bipartite b-matching. Here, we use a primal variable $m(s,r)$ for each edge $e = \{s,r\} \in E$, where $s \in S$ and $r \in R$, indicating whether or not e is contained in the matching.

$$\textbf{Primal:} \quad \max \sum_{\{s,r\} \in E} m(s,r)$$

$$\text{s.t.} \sum_{r:\{s,r\} \in E} m(s,r) \le b_s, \qquad (\forall s \in S)$$

$$\sum_{s:\{s,r\} \in E} m(s,r) \le 1, \qquad (\forall r \in R)$$

$$m(s,r) \ge 0, \qquad (\forall \{s,r\} \in E)\,.$$

$$\textbf{Dual:} \quad \min \sum_{s \in S} b_s \cdot x(s) + \sum_{r \in R} y(r)$$

$$\text{s.t. } x(s) + y(r) \ge 1, \qquad (\forall \{s,r\} \in E)$$

$$x(s),\, y(r) \ge 0, \qquad (\forall s \in S, \forall r \in R)\,.$$

Devanur et al. [6] developed an elegant framework that unifies the analysis of randomized online algorithms for matching problems. We will extend their framework to analyze our deterministic algorithm RelativeBalance. Whenever an online algorithm assigns a request r to a server s, the gain of 1 in the primal objective function (and thus the size of the matching) is translated into a gain of $1/c$ in the dual objective function by splitting it across the dual variables $x(s)$ and $y(r)$. Here, c is a constant that will be maximized during the analysis and will denote the competitive ratio of the algorithm, $0 < c \leq 1$. If an arriving request remains unmatched, the dual solution will remain unchanged as well.

It then has to be shown that this can be done in a way such that all the dual constraints are satisfied in the end. Let P and D be the value of the constructed primal and dual solution, respectively. By summing over all steps of the algorithm, we get $P = c \cdot D$, and thus $P \geq c \cdot \textsc{Opt}$, by weak duality. This implies that the online algorithm is c-competitive.

In the case without vertex capacities, Devanur et al. [6] show that for the known optimal randomized online algorithms that choose a random value $x_s \in [0, 1]$ for every server $s \in S$, the gain of matching a request r to s can be split across $x(s)$ and $y(r)$ according to the function $g(x_s) = e^{x_s - 1}$. More precisely, in the unweighted scenario, they argue that setting

$$x(s) = \frac{g(x_s)}{c} \text{ and } y(r) = \frac{1 - g(x_s)}{c}$$

with $c = 1 - 1/e$ results in a dual solution that is feasible in expectation.

In our case with vertex capacities, we first have to deal with the fact that a server s may be assigned multiple requests. Therefore, we increase the value of $x(s)$ whenever this happens. Moreover, we change the function that determines how the gain is split. Our algorithm uses the relative load of the servers for its matching decisions instead of a ranking based on the random values. Therefore, whenever RelativeBalance matches a request r to a server s, we update

$$\Delta x(s) = \frac{f(l_s)}{c \cdot b_s} \text{ and } y(r) = \frac{1 - f(l_s)}{c},$$

where $f : [0, 1] \to [0, 1]$ is a monotonically non-decreasing function and l_s denotes the relative load of s *before* the assignment. Observe that this increases the value of the dual solution by exactly $1/c$ and guarantees $x(s) \geq 0$ and $y(r) \geq 0$ for all $s \in S$ and $r \in R$, respectively.

Now, we have to show that f and c can be chosen such that this results in a feasible dual solution, i.e. $x(s) + y(r) \geq 1$ holds for all edges $\{s, r\} \in E$. If r is not matched by RelativeBalance, then $y(r) = 0$. Nonetheless, we know that all of r's neighbors had to be fully loaded when r arrived. Thus, in this case, for all b_s, we need that

$$x(s) + y(r) = \frac{1}{c \cdot b_s} \sum_{i=0}^{b_s - 1} f\left(\frac{i}{b_s}\right) \geq 1\,. \tag{1}$$

On the other hand, if r is matched to a server s' by RelativeBalance, then we know that $l_{s'} \leq l_s$ had to hold at the time of r's arrival. In this case, it therefore needs to hold that

$$x(s) + y(r) = \frac{1}{c}\left(\frac{1}{b_s} \sum_{i=0}^{load_s - 1} f\left(\frac{i}{b_s}\right) + 1 - f\left(\frac{load_{s'}}{b_{s'}}\right)\right) \geq 1\,, \tag{2}$$

for all ratios $load_{s'}/b_{s'} \leq load_s/b_s$. Recall that $load_{s'}$ and $load_s$ are absolute server loads.

▷ Claim 1. Let $c := 1 - 1/d$, where $d > 1$. Then, $f(l_s) = d^{l_s - 1}$ satisfies both (1) and (2) if $d \leq (1 + 1/b_s)^{b_s}$.

Proof. First, observe that $d \leq (1+1/b_s)^{b_s}$ implies $d^{\frac{1}{b_s}} - 1 \leq 1/b_s$. It then follows that

$$\frac{1}{c \cdot b_s} \sum_{i=0}^{b_s-1} f\left(\frac{i}{b_s}\right) = \frac{1}{c \cdot b_s \cdot d} \sum_{i=0}^{b_s-1} \left(d^{\frac{1}{b_s}}\right)^i = \frac{1}{c \cdot b_s \cdot d} \cdot \frac{d-1}{d^{\frac{1}{b_s}} - 1} \geq \frac{d-1}{c \cdot b_s \cdot d \cdot \frac{1}{b_s}} = 1\,.$$

The last step follows from the choice of c. Moreover, we can show that

$$\frac{1}{c}\left(\frac{1}{b_s} \sum_{i=0}^{load_s-1} f\left(\frac{i}{b_s}\right) + 1 - f\left(\frac{load_{s'}}{b_{s'}}\right)\right) = \frac{1}{c}\left(\frac{1}{b_s \cdot d} \sum_{i=0}^{load_s-1} \left(d^{\frac{1}{b_s}}\right)^i + 1 - d^{l_{s'}-1}\right)$$
$$= \frac{1}{c}\left(\frac{1}{b_s \cdot d} \cdot \frac{d^{l_s}-1}{d^{\frac{1}{b_s}}-1} + 1 - d^{l_{s'}-1}\right) \geq \frac{1}{c}\left(\frac{d^{l_s}-1}{d} + 1 - d^{l_{s'}-1}\right) \geq \frac{1}{c}\left(1 - \frac{1}{d}\right) = 1\,. \quad \triangleleft$$

We have now shown that the combination of $f(l_s) := d^{l_s-1}$ with $c = 1 - 1/d$ yields a feasible dual solution if $1 < d \leq (1+1/b_s)^{b_s}$, for all $s \in S$. Here c denotes the competitiveness of RelativeBalance. $(1+1/b_s)^{b_s}$ is a monotonically increasing function for $b_s > 0$. The largest possible value for d is therefore $(1+1/b_{\min})^{b_{\min}}$, where $b_{\min} = \min_s b_s$ is the smallest server capacity.

▶ **Theorem 2.** RelativeBalance *achieves a competitiveness of* $1 - 1/(1+1/b_{\min})^{b_{\min}}$, *where* $b_{\min} := \min_{s \in S} b_s$.

The competitive ratio of $1 - 1/(1+1/b_{\min})^{b_{\min}}$ is optimal for deterministic algorithms: Kalyanasudaram and Pruhs [13] showed that no deterministic online algorithm can achieve a competitiveness greater than $1 - 1/(1+1/b)^b$ if all servers have a uniform capacity equal to b. We can take their nemesis sequence and add servers with capacity $b' > b$ that are adjacent to few (or no) extra requests.

3 Randomized algorithms for maximum-cardinality b-matching

Karp et al. [15] proposed an algorithm Random, for online bipartite matching, which assigns a newly arriving request to a random eligible neighbor. They showed that Random is not better than $\frac{1}{2}$-competitive. We prove that the performance ratio does not improve, for the b-matching problem, even if all servers have a uniform capacity of $b \geq 2$. The material on Random with the proof of the following theorem is given in Appendix B.

▶ **Theorem 3.** Random *does not achieve a competitive ratio better than* $1/2$ *for the maximum cardinality online b-matching problem, even if all server capacities are equal.*

The remainder of this section is devoted to the Ranking algorithm. We will prove that the algorithm achieves an optimal competitiveness of $1 - 1/e$, for the maximum cardinality online b-matching problem. Again, we execute Ranking on the original input graph G. We will work with a version of Ranking (see Alg. 2) that is similar to that in [6]. Note that, importantly, there is a *single* random choice for each server $s \in S$. Initially, a $Z_s \in [0,1]$ is picked uniformly at random. This value is used as a rank for s. An incoming request is matched to the eligible server with smallest Z-value.

First, we argue that the classical primal-dual framework fails here, meaning that it is not able to establish a competitive ratio better than $1/2$. As usual, whenever Ranking assigns a request r to a server s, we increase $x(s)$ by and set $y(r)$ to

$$\Delta x(s) = \frac{g(Z_s)}{c \cdot b_s} \text{ and } y(r) = \frac{1 - g(Z_s)}{c}\,,$$

Algorithm 2 RANKING.

foreach *server* $s \in S$ **do**
  Pick $Z_s \in [0,1]$ uniformly at random;
end
while *a new request* $r \in R$ *arrives* **do**
  Let $N(r)$ denote the set of neighbors of r with remaining capacity;
  if $N(r) = \emptyset$ **then**
    Do not match r;
  else
    Match r to $\arg\min\{Z_s : s \in N(r)\}$ (break ties consistently);
  end
end

respectively. Again, $g : [0,1] \to [0,1]$ is a monotonically non-decreasing function and c is a constant that will denote the competitive ratio of the algorithm. Let P and D be random variables denoting the value of the random primal and dual solution, respectively. If we were able to show that a combination of g and c yields a dual solution that is feasible in expectation, then this would imply a competitive ratio of c. To see this, create a new (deterministic) dual solution that sets its variables to the expected value of the corresponding variable of the random solution and denote its value by D'. It then holds that the new dual solution satisfies all dual constraints and thus $\text{OPT} \leq D' = \mathbb{E}[D]$. Moreover, the framework yields $P = c \cdot D$, always, implying $\mathbb{E}[P] = c \cdot \mathbb{E}[D] \geq c \cdot \text{OPT}$.

Now, consider the two input graphs G_A and G_B (see Fig. 1). If there was a combination of g and c that always yields a dual solution that is feasible in expectation, then this combination also has to satisfy the constraint for the edge $\{s, r\}$ in both G_A and G_B in expectation. In graph G_A, this means

$$\mathbb{E}[x(s) + y(r)] = \frac{\mathbb{E}[g(Z_s)]}{c \cdot b_s} + \frac{1 - \mathbb{E}[g(Z_s)]}{c} \overset{!}{\geq} 1 \overset{b_s \geq 2}{\iff} \mathbb{E}[g(Z_s)] \leq (1-c) \cdot \frac{b_s}{b_s - 1}\,.$$

In G_B however, this means

$$\mathbb{E}[x(s) + y(r)] = b_s \cdot \frac{\mathbb{E}[g(Z_s)]}{c \cdot b_s} + 0 \overset{!}{\geq} 1 \iff \mathbb{E}[g(Z_s)] \geq c\,.$$

Combining these two inequalities yields $c \leq (1-c) \cdot \frac{b_s}{b_s - 1}$. This is equivalent to $c \leq \frac{b_s}{2b_s - 1}$, which implies that the best competitive ratio that may be shown for RANKING with this framework approaches $1/2$ for larger server capacities b_s.

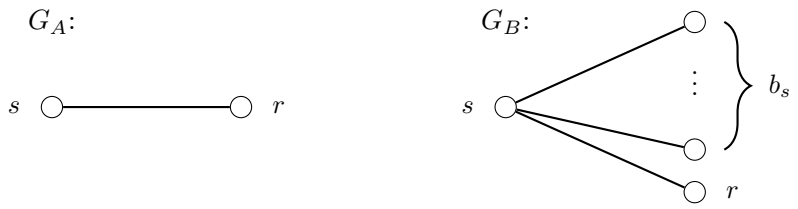

Figure 1 Two example input graphs for RANKING. Graph G_A only consists of a single edge between a server s and a request r, whereas G_B consists of a server s and its $b_s + 1$ neighboring requests. Request r denotes the last arriving request.

Given this fact, we proceed and model the b-matching problem by a configuration LP. Let N_s denote the set of neighbors of a server s. The configuration LP differs from the classical matching LP in that it does not use a variable for every edge $\{s, r\}$ indicating whether this edge is chosen by the algorithm. Instead it uses a variable $m(s, N)$, for every server s and every subset $N \subseteq N_s$, indicating whether this subset is the set of requests matched to s.

$$\textbf{Config LP: } \max \sum_{s \in S} \sum_{N \subseteq N_s} \min\{|N|, b_s\} \cdot m(s, N)$$

$$\text{s.t.} \sum_{N \subseteq N_s} m(s, N) \leq 1, \qquad (\forall s \in S)$$

$$\sum_{s \in S} \sum_{N \subseteq N_s : r \in N} m(s, N) \leq 1, \qquad (\forall r \in R)$$

$$m(s, N) \geq 0, \qquad (\forall s \in S, \forall N \subseteq N_s)\,.$$

$$\textbf{Dual CLP: } \min \sum_{s \in S} x(s) + \sum_{r \in R} y(r)$$

$$\text{s.t. } x(s) + \sum_{r \in N} y(r) \geq \min\{|N|, b_s\}, \qquad (\forall s \in S, \forall N \subseteq N_s)$$

$$x(s), y(r) \geq 0, \qquad (\forall s \in S, \forall r \in R)\,.$$

Obviously, every valid b-matching in a graph G is captured by a solution of the configuration LP. Its optimal solution is an upper bound on the cardinality of the maximum b-matching in G. Hence the configuration LP is a suitable primal program for a primal-dual analysis.

We adapt the primal-dual analysis framework. Initially, all primal and dual variables are set to 0. Whenever a new request $r \in R$ arrives and Ranking assigns it to a server s, we update the primal variables of s accordingly, keeping track of the set N of matching partners. The value of the primal solution increases by 1. Moreover, we update the dual variables

$$\Delta x(s) = \frac{g(Z_s)}{c} \text{ and } y(r) = \frac{1 - g(Z_s)}{c}\,,$$

where $g : [0, 1] \to [0, 1]$ is a monotonically non-decreasing function to be determined during the analysis and c is a constant that will denote the competitive ratio of the algorithm. Note that we now do not have to divide the gain of $x(s)$ by b_s, since the dual objective function does not have a factor b_s before $x(s)$. Therefore, we still translate a gain of 1 in the primal solution to a gain of $1/c$ in the dual solution, guaranteeing that $P = c \cdot D$, where P and D are the random variables denoting the value of the primal and dual solution, respectively. Similar arguments to before imply that it is sufficient to satisfy all dual constraints in expectation to show a competitive ratio of c.

Thus, it remains to show is that we can choose g and c such that

$$\mathbb{E}\left[x(s) + \sum_{r \in N} y(r)\right] \geq \min\{|N|, b_s\}\,,$$

for all servers $s \in S$ and all $N \subseteq N_s$. For this, we will need two lemmas similar to the Dominance and Monotonicity Lemmas in [6]. We will consider two executions of Ranking on G and on $G \setminus s$, for some server $s \in S$. Here, $G \setminus s$ denotes the graph induced by the vertex set $S \setminus \{s\} \cup R$. We assume that Ranking uses the same Z-values Z_t for all servers $t \in S \setminus \{s\}$ in both executions. Further, let $r \in R$ be any request in G and let z_r be the

Z-value of its matching partner in the $G \setminus s$ execution. If r is unmatched, we set $z_r := 1$ and assign a dummy matching partner. Moreover, let $y_s(r)$ be the value of $y(r)$ in the $G \setminus s$ execution. We impose from now on that $g(1) = 1$, which implies $y_s(r) = (1 - g(z_r))/c$. We use the idea of *server spots*. A server spot s_i of a server s, $1 \leq i \leq b_s$, denotes an individual unit of a server that can accept a request. When RANKING assigns requests to a server s, we assume without loss of generality that it assigns the j-th request to the server spot s_j. A server spot is matched if it has been assigned a request, and unmatched otherwise.

▶ **Lemma 4.** *At any point during the parallel execution of* RANKING *on G and $G \setminus s$, the set of unmatched server spots U in the G execution forms a superset of the unmatched server spots $\tilde{U}$ in the $G \setminus s$ execution. For all server spots $s'_i \in U \setminus \tilde{U}$, it holds that $Z_{s'} \geq Z_s$. If $Z_{s'} = Z_s$ and $s' \neq s$, then s has a higher priority in the tiebreaking.*

Proof. By induction. Initially, the properties trivially hold, since $U \setminus \tilde{U} = \{s_1, \ldots, s_{b_s}\}$ at the start. Then, whenever a new request r arrives, $\tilde{U} \subseteq U$ can only be violated if r is assigned to a server spot $t_i \in \tilde{U}$ in the G execution, but r is not assigned to t_i in the $G \setminus s$ execution. There, it is either unmatched or matched to a different server spot, which leads to a contradiction in either case. Since $t_i \in \tilde{U}$ and r is a neighbor of the server t, r cannot be unmatched in the $G \setminus s$ execution. If RANKING chooses a different server spot t_j for r in the $G \setminus s$ execution, then either $i < j$ or $i > j$ has to hold. $i < j$ results in a contradiction because $t_i \in \tilde{U}$ and we defined that RANKING always chooses the unmatched server spot with smallest index. Furthermore, $i > j$ also results in a contradiction because $t_j \in \tilde{U} \subseteq U$ and thus RANKING would have chosen t_j in the G execution as well. Moreover, if RANKING assigns r to a server spot of a different server t' in the $G \setminus s$ execution, then $Z_{t'} \leq Z_t$ has to hold. However, $\tilde{U} \subseteq U$ implies that this server spot would also be unmatched and available in the G execution. If $Z_{t'} < Z_t$, RANKING would not have chosen an unmatched neighbor with smallest Z-value in the G execution, and if $Z_{t'} = Z_t$, then the tiebreak would be inconsistent between the two executions.

Moreover, a new server spot t'_i is only added to $U \setminus \tilde{U}$ if the matching decision for r is different in the two execution, i.e. the G execution assigns r to some server spot $t_j \in U \setminus \tilde{U}$ and the $G \setminus s$ execution assigns r to $t'_i \in \tilde{U}$. Therefore, it has to hold that either $Z_{t'} > Z_t$ or $Z_{t'} = Z_t$ and t has a higher tiebreak priority than t'. By induction hypothesis, $Z_t \geq Z_s$ and thus $Z_{t'} \geq Z_s$. If $t' \neq s$ and $Z_{t'} = Z_t = Z_s$, then by induction hypothesis s has a higher tiebreak priority than t, which in turn has a higher priority than t'. We conclude that s has a higher tiebreak priority than t'. ◀

Hence, if a request r is unmatched in the G execution, it is also unmatched in the $G \setminus s$ execution. If r gets matched, its matching partner has a Z-value of at most z_r. Since g is non-decreasing with $g(1) = 1$, the following statement holds.

▶ **Corollary 5.** *Given Z_t for all servers $t \in S \setminus \{s\}$, $y(r) \geq y_s(r)$ holds for all possible values of Z_s.*

▶ **Lemma 6.** *Given Z_t for all servers $t \in S \setminus \{s\}$, let $z_1 \geq \ldots \geq z_k$ be the Z-values of the matching partners of the $k = |N_s|$ neighbors of s in a $G \setminus s$ execution in non-increasing order. Then, server s has at least $\min\{a, b_s\}$ matching partners in an execution of* RANKING *on G, where a is the largest possible integer such that $Z_s < z_a \leq \ldots \leq z_1$.*

Proof. Whenever a neighbor r_i of s with $z_i > Z_s$ arrives and s still has remaining capacity, then by Lemma 4 r_i will be matched to s. Among adjacent servers with remaining capacity, s has the smallest Z-value and the highest priority in case of ties. ◀

Now, we can finally show how to choose g and c such that the dual constraints are satisfied in expectation. Let s be any server in G with k neighbors. Let z_i be the Z-value of the matching partner of neighbor $r_i \in N_s$, $1 \leq i \leq k$, in the $G \setminus s$ execution. If $k < b_s$, we further define $z_{k+1} = \ldots = z_{b_s} = 0$. Let $z'_1 \geq \ldots \geq z'_{b_s}$ then be the b_s largest values of $\{z_1, \ldots, z_{\max\{k,b_s\}}\}$ in non-increasing order. Lemma 6 implies that

$$\mathbb{E}\left[x(s) \,\middle|\, \bigwedge_{t \in S\setminus\{s\}} Z_t = z_t\right] \geq \sum_{i=1}^{b_s} \int_0^{z'_i} \frac{g(t)}{c}\,\mathrm{d}t\,.$$

Moreover, by Corollary 5, it holds for every neighbor $r \in N_s$ of s

$$\mathbb{E}\left[y(r) \,\middle|\, \bigwedge_{t \in S\setminus\{s\}} Z_t = z_t\right] \geq y_s(r) = \frac{1 - g\left(z_r\right)}{c}\,,$$

where $z_r = z_i$ for some i, $1 \leq i \leq k$. Putting everything together yields

$$\mathbb{E}\left[x(s) + \sum_{r \in N} y(r) \,\middle|\, \bigwedge_{t \in S\setminus\{s\}} Z_t = z_t\right] \geq \frac{1}{c}\left(\sum_{i=1}^{b_s} \int_0^{z'_i} g(t)\,\mathrm{d}t + \sum_{r \in N}\left(1 - g\left(z_r\right)\right)\right)\,.$$

Note that $\sum_{r \in N}\left(1 - g\left(z_r\right)\right)$ is lower bounded by $\sum_{i=1}^{\min\{|N|,b_s\}}\left(1 - g\left(z'_i\right)\right)$, since g is a non-decreasing function with $g(1) = 1$ and the z'-values are an upper bound for the z_r-values. Plugging this in, we get

$$\begin{aligned}\mathbb{E}\left[x(s) + \sum_{r \in N} y(r) \,\middle|\, \bigwedge_{t \in S\setminus\{s\}} Z_t = z_t\right] &\geq \frac{1}{c}\sum_{i=1}^{\min\{|N|,b_s\}}\left(\int_0^{z'_i} g(t)\,\mathrm{d}t + 1 - g\left(z'_i\right)\right)\\ &\overset{!}{\geq} \min\{|N|, b_s\}\,.\end{aligned}$$

Observe that the last inequality holds if g and c satisfy the following inequality, which is the same inequality that emerges in the analysis of Ranking without server capacities.

$$\int_0^z g(t)\,\mathrm{d}t + 1 - g\left(z\right) \geq c\,, \quad \forall z \in [0,1]\,. \tag{3}$$

It is easy to check that the combination of $g(x) = e^{x-1}$ with $c = 1 - 1/e$ satisfies (3) and our additional condition $g(1) = 1$. By applying the law of total expectation, we finish the proof:

$$\begin{aligned}\mathbb{E}\left[x(s) + \sum_{r \in N} y(r)\right] &= \int_0^1 \ldots \int_0^1 \mathbb{E}\left[x(s) + \sum_{r \in N} y(r) \,\middle|\, \bigwedge_{t \in S\setminus\{s\}} Z_t = z_t\right] \mathrm{d}z_t \ldots \mathrm{d}z_{t'}\\ &\geq \int_0^1 \ldots \int_0^1 \min\{|N|, b_s\}\,\mathrm{d}z_t \ldots \mathrm{d}z_{t'} = \min\{|N|, b_s\}\,.\end{aligned}$$

▶ **Theorem 7.** Ranking *is* $(1 - 1/e)$*-competitive for the maximum-cardinality online b-matching problem (with variable server capacities).*

4 Vertex-weighted b-matching

The work by Buchbinder et al. [4] implies a deterministic online algorithm with an optimal competitiveness of $1-1/(1+1/b_{\min})^{b_{\min}}$. We are not aware of any simpler strategy. Therefore, we focus on randomized algorithms and extend our previous result for Ranking to the vertex-weighted case. We will show that Perturbed-Greedy [1] achieves a competitiveness of $1-1/e$ for vertex-weighted b-matching. Again, we execute the algorithm on the initial input graph G.

Perturbed-Greedy is similar to Ranking; only the definition of ranks differs. For each server $s \in S$, a *single* number $Z_s \in [0,1]$ is chosen uniformly at random. The rank of s is $w_s(1-g(Z_s))$, where $g:[0,1]\to[0,1]$ is a monotonically increasing function that will be set to $g(x)=e^{x-1}$.

Algorithm 3 Perturbed-Greedy.

foreach *server* $s \in S$ **do**
  Pick $Z_s \in [0,1]$ uniformly at random;
end
while *a new request* $r \in R$ *arrives* **do**
  Let $N(r)$ denote the set of neighbors of r with remaining capacity;
  if $N(r)=\emptyset$ **then**
    Do not match r;
  else
    Match r to $\arg\max\{w_s\,(1-g\,(Z_s)) : s \in N(r)\}$ (break ties consistently);
  end
end

We formulate the configuration LP and its dual for the vertex-weighted b-matching problem, where we take into account that each matching edge incident to a server s has a value of w_s.

$$\textbf{Config LP: } \max \sum_{s\in S}\sum_{N\subseteq N_s} w_s \cdot \min\{|N|, b_s\} \cdot m(s,N)$$

$$\text{s.t.} \sum_{N\subseteq N_s} m(s,N) \le 1, \qquad (\forall s \in S)$$

$$\sum_{s\in S}\sum_{N\subseteq N_s : r\in N} m(s,N) \le 1, \qquad (\forall r \in R)$$

$$m(s,N) \ge 0, \qquad (\forall s \in S, \forall N \subseteq N_s)\,.$$

$$\textbf{Dual CLP: } \min \sum_{s\in S} x(s) + \sum_{r\in R} y(r)$$

$$\text{s.t. } x(s) + \sum_{r\in N} y(r) \ge w_s \cdot \min\{|N|, b_s\}, \qquad (\forall s \in S, \forall N \subseteq N_s)$$

$$x(s), y(r) \ge 0, \qquad (\forall s \in S, \forall r \in R)\,.$$

In the primal-dual analysis, we again update the primal variables as well as the dual variables $x(s)$ and $y(r)$ whenever Perturbed-Greedy matches a request r to a server s. We set

$$\Delta x(s) = \frac{w_s g(Z_s)}{c} \text{ and } y(r) = \frac{w_s\,(1-g(Z_s))}{c}$$

to ensure that the value of the dual solution is always $1/c$ times the value of the solution for the configuration LP. Here, $g : [0,1] \to [0,1]$ is a monotonically increasing function and c is a constant that will be the competitive ratio of the algorithm.

As before, it is sufficient to show that the dual constraints are satisfied in expectation. For this, we have to adapt Lemma 4 and Lemma 6. We consider two execution of Perturbed-Greedy on G and $G \setminus s$ with the same Z-values Z_t for all servers $t \in S \setminus \{s\}$. Moreover, denote the neighbors of s in G by $\{r_1, \ldots, r_k\} = N_s$ and let z_i, $1 \leq i \leq k$, be the Z-value of the matching partner σ_i of request r_i in the $G \setminus s$ execution, if r_i is matched there. If r_i is unmatched, we set $z_r := 1$ and assign a dummy matching partner σ_i with $w_{\sigma_i} := 0$. Now, further define ζ_i as the unique value in $[0,1]$ such that

$$w_s\left(1 - g\left(\zeta_i\right)\right) = w_{\sigma_i}\left(1 - g\left(z_i\right)\right) ,$$

if it exists. Assuming that g is a monotonically increasing function with $g(1) = 1$, note that such a solution can only not exist if $w_{\sigma_i}\left(1 - g\left(z_i\right)\right) > w_s\left(1 - g\left(0\right)\right)$, in which case we define $\zeta_i := 0$. It is easy to see that Perturbed-Greedy would prefer server s over server σ_i if $Z_s < \zeta_i$. Moreover, let $y_s(r_i)$ be the value of $y(r_i)$ in the $G \setminus s$ execution. It follows that

$$y_s(r_i) = \frac{w_{\sigma_i}\left(1 - g\left(z_i\right)\right)}{c} \geq \frac{w_s\left(1 - g\left(\zeta_i\right)\right)}{c} .$$

We assume that Perturbed-Greedy assigns the j-th request matched to a server s to the server spot s_j.

▶ **Lemma 8.** *At any point during the parallel execution of* Perturbed-Greedy *on G and $G \setminus s$, it holds that the set of unmatched server spots U in the G execution forms a superset of the unmatched server spots $\tilde{U}$ in the $G \setminus s$ execution. For all server spots $s'_i \in U \setminus \tilde{U}$, it holds that $w_{s'}\left(1 - g\left(Z_{s'}\right)\right) \leq w_s\left(1 - g\left(Z_s\right)\right)$. If equality holds and $s' \neq s$, then s has a higher priority in the tiebreaking.*

Proof. By induction. The properties trivially hold initially. Then, whenever a new request r arrives, $\tilde{U} \subseteq U$ can only be violated if r is assigned to a server spot $t_i \in \tilde{U}$ in the G execution, but r is not assigned to t_i in the $G \setminus s$ execution. There, it is either unmatched or matched to a different server spot, which leads to a contradiction in either case. Since $t_i \in \tilde{U}$ and r is a neighbor of the server t, r cannot be unmatched in the $G \setminus s$ execution. If Perturbed-Greedy chooses a different server spot t_j for r in the $G \setminus s$ execution, then either $i < j$ or $i > j$ has to hold. $i < j$ results in a contradiction because $t_i \in \tilde{U}$ and we defined that Perturbed-Greedy always chooses the server spot with smallest index. Furthermore, $i > j$ also results in a contradiction because $t_j \in \tilde{U} \subseteq U$ and thus Perturbed-Greedy would have chosen t_j in the G execution as well. Moreover, if Perturbed-Greedy assigns r to a server spot of a different server t' in the $G \setminus s$ execution, then $w_{t'}\left(1 - g\left(Z_{t'}\right)\right) \geq w_t\left(1 - g\left(Z_t\right)\right)$ has to hold. However, $\tilde{U} \subseteq U$ implies that this server spot would also be unmatched and available in the G execution. If $w_{t'}\left(1 - g\left(Z_{t'}\right)\right) > w_t\left(1 - g\left(Z_t\right)\right)$, Perturbed-Greedy would not have chosen the correct neighbor in the G execution according to its definition, and if $w_{t'}\left(1 - g\left(Z_{t'}\right)\right) = w_t\left(1 - g\left(Z_t\right)\right)$, then the tiebreak would be inconsistent between the two executions.

Moreover, a new server spot t'_i is only added to $U \setminus \tilde{U}$ if the matching decision for r is different in the two execution, i.e. the G execution assigns r to some server spot $t_j \in U \setminus \tilde{U}$ and the $G \setminus s$ execution assigns r to $t'_i \in \tilde{U}$. Therefore, it has to hold that either $w_{t'}\left(1 - g\left(Z_{t'}\right)\right) < w_t\left(1 - g\left(Z_t\right)\right)$ or $w_{t'}\left(1 - g\left(Z_{t'}\right)\right) = w_t\left(1 - g\left(Z_t\right)\right)$ and t has a higher tiebreak priority than t'. The induction hypothesis then finishes the proof. ◀

▶ **Corollary 9.** *Given Z_t for all servers $t \in S \setminus \{s\}$, $y(r_i) \geq y_s(r_i) \geq w_s\left(1 - g\left(\zeta_i\right)\right)/c$ holds for all i, $1 \leq i \leq k$, and all possible values of Z_s.*

▶ **Lemma 10.** *Given Z_t for all servers $t \in S \setminus \{s\}$, let $\zeta_1 \geq \ldots \geq \zeta_k$ be the ζ-values of the $k = |N_s|$ neighbors of s in a $G \setminus s$ execution in non-increasing order. Then, server s has at least $\min\{a, b_s\}$ matching partners in an execution of* Perturbed-Greedy *on G, where a is the largest possible integer such that $Z_s < \zeta_a \leq \ldots \leq \zeta_1$.*

Proof. Whenever a neighbor r_i of s with $\zeta_i > Z_s$ (note that this implies $\zeta_i > 0$) arrives and s still has remaining capacity, then Lemma 8 implies that r_i will be matched to s since $w_s\left(1 - g\left(Z_s\right)\right) > w_s\left(1 - g\left(\zeta_i\right)\right) = w_{\sigma_i}\left(1 - g\left(z_i\right)\right)$ holds by definition. ◀

We finally show how to choose g and c such that the dual constraints are satisfied in expectation. Let s be any server in G with k neighbors. Let ζ_i be the ζ-value of neighbor $r_i \in N_s$, $1 \leq i \leq k$, in the $G \setminus s$ execution. If $k < b_s$, we further define $\zeta_{k+1} = \ldots = \zeta_{b_s} = 0$. Let $z'_1 \geq \ldots \geq z'_{b_s}$ then be the b_s largest values of $\{\zeta_1, \ldots, \zeta_{\max\{k, b_s\}}\}$ in non-increasing order. Lemma 10 implies that

$$\mathbb{E}\left[x(s) \,\middle|\, \bigwedge_{t \in S \setminus \{s\}} Z_t = z_t\right] = \sum_{i=1}^{b_s} w_s \int_0^{z'_i} \frac{g(t)}{c} \,\mathrm{d}t\,.$$

Moreover, by Corollary 9, it holds for every neighbor $r \in N_s$ of s

$$\mathbb{E}\left[y(r) \,\middle|\, \bigwedge_{t \in S \setminus \{s\}} Z_t = z_t\right] \geq \frac{w_s\left(1 - g\left(\zeta_r\right)\right)}{c}\,,$$

where $\zeta_r = \zeta_i$ for some i, $1 \leq i \leq k$. Putting everything together yields

$$\mathbb{E}\left[x(s) + \sum_{r \in N} y(r) \,\middle|\, \bigwedge_{t \in S \setminus \{s\}} Z_t = z_t\right] \geq \frac{w_s}{c}\left(\sum_{i=1}^{b_s} \int_0^{z'_i} g(t)\,\mathrm{d}t + \sum_{r \in N}\left(1 - g\left(\zeta_r\right)\right)\right).$$

Note that $\sum_{r \in N}\left(1 - g\left(\zeta_r\right)\right)$ is lower bounded by $\sum_{i=1}^{\min\{|N|, b_s\}}\left(1 - g\left(z'_i\right)\right)$, since g is an increasing function with $g(1) = 1$ and the z'-values are an upper bound for the ζ_r-values. Plugging this in, we get

$$\begin{aligned}\mathbb{E}\left[x(s) + \sum_{r \in N} y(r) \,\middle|\, \bigwedge_{t \in S \setminus \{s\}} Z_t = z_t\right] &\geq \frac{w_s}{c} \sum_{i=1}^{\min\{|N|, b_s\}} \left(\int_0^{z'_i} g(t)\,\mathrm{d}t + 1 - g\left(z'_i\right)\right) \\ &\overset{!}{\geq} w_s \min\{|N|, b_s\}\,.\end{aligned}$$

Observe that this holds true if g and c fulfill the same inequality (3) as for Ranking. As argued before, it is satisfied for $g(x) = e^{x-1}$ and $c = 1 - 1/e$, which also satisfies our additional constraint $g(1) = 1$. Therefore, using the law of total expectation, we conclude that

$$\begin{aligned}\mathbb{E}\left[x(s) + \sum_{r \in N} y(r)\right] &= \int_0^1 \ldots \int_0^1 \mathbb{E}\left[x(s) + \sum_{r \in N} y(r) \,\middle|\, \bigwedge_{t \in S \setminus \{s\}} Z_t = z_t\right] \mathrm{d}z_t \ldots \mathrm{d}z_{t'} \\ &\geq \int_0^1 \ldots \int_0^1 w_s \min\{|N|, b_s\}\,\mathrm{d}z_t \ldots \mathrm{d}z_{t'} = w_s \min\{|N|, b_s\}\,.\end{aligned}$$

▶ **Theorem 11.** Perturbed-Greedy *is $(1 - 1/e)$-competitive for the vertex-weighted online b-matching problem (with variable server capacities).*

References

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

2 Y. Azar and A. Litichevskey. Maximizing throughput in multi-queue switches. *Algorithmica*, 45(1):69–90, 2006.

3 B.E. Birnbaum and C. Mathieu. On-line bipartite matching made simple. *SIGACT News*, 39(1):80–87, 2008.

4 N. Buchbinder, K. Jain, and J. Naor. Online primal-dual algorithms for maximizing ad-auctions revenue. In *Proceedings of the 15th Annual European Symposium on Algorithms (ESA)*, volume 4698 of *Lecture Notes in Computer Science*, pages 253–264. Springer, 2007.

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

6 N.R. Devanur, K. Jain, and R.D. Kleinberg. Randomized primal-dual analysis of RANKING for online bipartite matching. In *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 101–107, 2013.

7 A. Eden, M. Feldman, A. Fiat, and K. Segal. An economics-based analysis of RANKING for online bipartite matching. In *Proceedings of the 4th Symposium on Simplicity in Algorithms (SOSA)*, pages 107–110, 2021.

8 G. Goel and A. Mehta. Online budgeted matching in random input models with applications to adwords. In *Proceedings of the 19thAnnual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 982–991, 2008.

9 E.F. Grove, M.-Y. Kao, P. Krishnan, and J.S. Vitter. Online perfect matching and mobile computing. In *Proceedings 4th International Workshop, on Algorithms and Data Structures (WADS)*, volume 955 of *Lecture Notes in Computer Science*, pages 194–205. Springer, 1995.

10 Z. Huang and Q. Zhang. Online primal dual meets online matching with stochastic rewards: configuration LP to the rescue. In *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 1153–1164, 2020.

11 Z. Huang, Q. Zhang, and Y. Zhang. Adwords in a panorama. In *Proceedings of the 61st IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1416–1426, 2020.

12 B. Jin and D.P. Williamson. Improved analysis of RANKING for online vertex-weighted bipartite matching. *CoRR*, abs/2007.12823, 2020. `arXiv:2007.12823`.

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

14 C. Karande, A. Mehta, and P. Tripathi. Online bipartite matching with unknown distributions. In *Proceedings of the 43rd ACM Symposium on Theory of Computing (STOC)*, pages 587–596. ACM, 2011.

15 R.M. Karp, U.V. Vazirani, and V.V. Vazirani. An optimal algorithm for on-line bipartite matching. In *Proceedings of the 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 *Proceedings of the 43rd ACM Symposium on Theory of Computing (STOC)*, pages 597–606, 2011.

17 A. Mehta and D. Panigrahi. Online matching with stochastic rewards. In *53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 728–737, 2012.

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

A Comparison of RelativeBalance and Allocation

Consider the algorithm Allocation [4] specialized for maximum cardinality online b-matching. It is the other known optimal deterministic online algorithm for this problem besides RelativeBalance.

Algorithm 4 Allocation.

Initialize $x(s) = 0$, $\forall s \in S$, and $y(r) = 0$, $\forall r \in R$;
while *a new request* $r \in R$ *arrives* **do**
  Let $N(r)$ denote the set of neighbors s of r with $x(s) < 1$;
  if $N(r) = \emptyset$ **then**
    Do not match r;
  else
    Match r to $\arg\min\{x(s) : s \in N(r)\}$ (break ties arbitrarily);
    Upadte $y(r) = 1 - x(s)$;
    Update $x(s) = x(s) \cdot \left(1 + \frac{1}{b_s}\right) + \frac{1}{(d-1)b_s}$;
  end
end

The analysis of Buchbinder et al. [4] can be extended to show that Allocation constructs a feasible solution for the classical dual matching LP with its variables $x(s)$ and $y(r)$. Moreover, it can be shown that the size of the constructed matching is at least $c = 1 - 1/d$ times the value of the constructed dual solution, where $d = (1 + 1/b_{\min})^{b_{\min}}$. This implies that Allocation achieves the optimal competitiveness of $1 - 1/(1 + 1/b_{\min})^{b_{\min}}$.

Recall that RelativeBalance matches a request r to an eligible neighbor with minimum relative server load. In contrast, Allocation matches r to a neighbor s with minimum $x(s)$, if $x(s) < 1$. It can be proven by induction that at any point during the execution of Allocation, a server s with capacity b_s and $load_s$ assigned requests has

$$x(s) = \frac{1}{d-1}\left(\left(1 + \frac{1}{b_s}\right)^{load_s} - 1\right). \tag{4}$$

This has two consequences: on the one hand, Allocation may choose a different matching partner for r compared to RelativeBalance in certain situations, since $load_s/b_s \leq load_{s'}/b_{s'}$ does not imply $(1 + 1/b_s)^{load_s} \leq (1 + 1/b_{s'})^{load_{s'}}$. On the other hand, Allocation considers s to be full once $x(s) \geq 1$. Equation (4) implies that this is the case when

$$\left(1 + \frac{1}{b_s}\right)^{load_s} \geq \left(1 + \frac{1}{b_{\min}}\right)^{b_{\min}}.$$

Observe that this may occur before $load_s$ becomes b_s, meaning before s actually has been assigned b_s requests. This implies that an unmodified version of Allocation may leave some server spots unused and thus not create a maximal matching.

B Analysis of Random

We start with a pseudo-code description of Random.

Algorithm 5 Random.

```
while a new request r ∈ R arrives do
    Let N(r) denote the set of neighbors of r with remaining capacity;
    if N(r) = ∅ then
        Do not match r;
    else
        Match r to a random s ∈ N(r);
    end
end
```

We extend Random's worst case input graph of the problem without server capacities and show that Random also does not achieve a competitive ratio better than $\frac{1}{2}$ for the online b-matching problem, even if all server capacities are equal. Consider a graph with n servers $S = \{s_1, \ldots, s_n\}$ and n *rounds* of requests $R = R_1 \dot\cup \ldots \dot\cup R_n$, where $n = 2k$. Every server s has the same capacity $b_s := b$ and each round contains b identical requests that all have the same neighbors. The different rounds arrive one after another, such that the first request of R_{i+1} only arrives after the last request of R_i arrived, $1 \leq i < n$. Requests within the same round can arrive in an arbitrary order. All requests $r \in R_i$ of the i-th round are adjacent to server s_i. This implies that there exists a perfect matching of size $b \cdot n$ that matches R_i to s_i. Moreover, all requests $r \in R_1 \dot\cup \ldots \dot\cup R_k$ of the first half of rounds are additionally adjacent to all servers $s_{k+1}, \ldots, s_n$ of the second half (see Fig. 2).

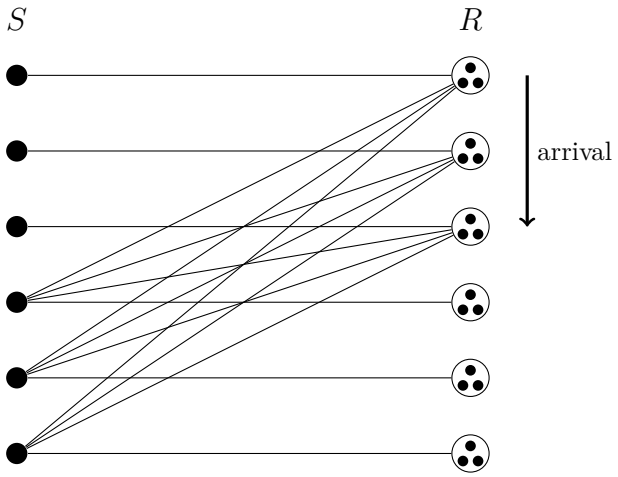

Figure 2 A bad input for Random. There are $n = 6$ servers, each with capacity $b = 3$, and n rounds of requests, each containing b identical requests. For clarity, the adjacencies of a round are depicted as a whole. Note that requests still arrive individually one after another and not together with their complete round.

Intuitively, Random performs poorly on this graph since its very unlikely that it makes the correct matching decision for the requests from the first half or rounds, i.e. assigning a request from R_i to server s_i. Observe that - irrespective of the matching decision made

by RANDOM for the requests of the first half of rounds - every server of the second half will be assigned exactly b requests. Let X be a random variable indicating how many requests were matched to the first half of servers by RANDOM. The size of the constructed matching M is then $|M| = b \cdot k + X$. Therefore, it is possible to compute the expected size of the constructed matching by determining the expected value of X.

Let X_i, $1 \leq i \leq k$, be the number of requests assigned to server s_i. It holds that $X = \sum_{i=1}^{k} X_i$. By design, only requests from R_i may be assigned to s_i, for $1 \leq i \leq k$. Let r be any request from such a round R_i and let p_i be the probability that RANDOM assigns r to its perfect matching partner s_i. Observe that p_i depends on the number of servers in the second half with remaining capacity. At most $b \cdot (i-1) + (b-1)$ requests arrived before r (r may be the last request of R_i). Hence at most $(i-1)$ of the last k servers can be full, implying at least $(k-i+1)$ eligible neighbors in the second half of servers for all requests from R_i. Furthermore, server s_i cannot become full before the last request of R_i arrives. RANDOM has therefore at least $(k-i+2)$ servers to choose from when assigning a request from round R_i. This yields

$$\mathbb{E}[X] = \sum_{i=1}^{k} \mathbb{E}[X_i] \leq \sum_{i=1}^{k} b \cdot p_i \leq b \sum_{i=1}^{k} \frac{1}{k+2-i} = b \sum_{j=2}^{k+1} \frac{1}{j} = b\,(H_{k+1} - 1) \leq b \ln(k+1)\,,$$

where H_n denotes the n-th harmonic number and the inequality $H_n \leq \ln(n) + 1$ is used. The size of the perfect matching in this graph is $b \cdot n$. Thus, RANDOM achieves a competitive ratio of

$$\frac{\mathbb{E}[|M|]}{b \cdot n} \leq \frac{b \cdot k + b \ln(k+1)}{b \cdot n} = \frac{1}{2} + \frac{\ln(n/2+1)}{n} \xrightarrow{n \to \infty} \frac{1}{2}\,.$$

This finishes the proof of Theorem 3.