Simultaneous Approximations for Adversarial and Stochastic Online Budgeted Allocation

Vahab S. Mirrokni [*] Shayan Oveis Gharan[‡†] Morteza Zadimoghaddam [‡§]

September 29, 2011

Abstract

Motivated by applications in online ad allocation, we study the problem of simultaneous approximations for the adversarial and stochastic online budgeted allocation problem. This problem consists of a bipartite graph $G = (X, Y, E)$, where the nodes of Y along with their corresponding capacities are known beforehand to the algorithm, and the nodes of X arrive online. When a node of X arrives, its incident edges, and their respective weights are revealed, and the algorithm can match it to a neighbor in Y. The objective is to maximize the weight of the final matching, while respecting the capacities.

When nodes arrive in an adversarial order, the best competitive ratio is known to be $1 - 1/e$, and it can be achieved by the Ranking [16], and its generalizations (Balance [14, 19]). On the other hand, if the nodes arrive through a random permutation, it is possible to achieve a competitive ratio of $1 - \epsilon$ [9]. In this paper we design algorithms that achieve a competitive ratio better than $1 - 1/e$ on average, while preserving a nearly optimal worst case competitive ratio. Ideally, we want to achieve the best of both worlds, i.e, to design an algorithm with the optimal competitive ratio in both the adversarial and random arrival models. We achieve this for unweighted graphs, but show that it is not possible for weighted graphs.

In particular, for unweighted graphs, under some mild assumptions, we show that Balance achieves a competitive ratio of $1 - \epsilon$ in a random permutation model. For weighted graphs, however, we show that this is not possible; we prove that no online algorithm that achieves an approximation factor of $1 - \frac{1}{e}$ for the worst-case inputs may achieve an average approximation factor better than 97.6% for random inputs. In light of this hardness result, we aim to design algorithms with improved approximation ratios in the random arrival model while preserving the competitive ratio of $1 - \frac{1}{e}$ in the worst case. To this end, we show the algorithm proposed by [19] achieves a competitive ratio of 0.76 for the random arrival model, while having a $1 - \frac{1}{e}$ ratio in the worst case.

1 Introduction

Online bipartite matching is a fundamental optimization problem with many applications in online resource allocation, especially the online allocation of ads on the Internet. In this problem, we are

[*]Google Research, 76 9th Ave, New York, NY 10011, Email:mirrokni@google.com.

[†]Department of Management Science and Engineering, Stanford University. Supported by a Stanford Graduate Fellowship. Email:shayan@stanford.edu.

[‡]The work was done while the author was a summer intern at Microsoft Research New England.

[§]MIT Computer Science and Artificial Intelligence Laboratory, Cambridge, MA 02139, USA. Email:morteza@mit.edu.

given a bipartite graph $G = (X, Y, E)$ with a set of fixed nodes (or bins) Y, a set of online nodes (or balls) X, and a set E of edges between them. Any fixed node (or bin) $y_j \in Y$ is associated with a total weighted capacity (or budget) c_j. Online nodes (or balls) $x_i \in X$ arrive online along with their incident edges $(x_i, y_j) \in E(G)$ and their weights $w_{i,j}$. Upon the arrival of a node $x_i \in X$, the algorithm can assign x_i to at most one bin $y_j \in Y$ where $(x_i, y_j) \in E(G)$ and the total weight of nodes assigned to y_j does not exceed c_j. The goal is to maximize the total weight of the allocation. This problem is known as the *AdWords* problem, and it has been studied under the assumption that $\frac{\max_{i,j} w_{i,j}}{\min_j c_j} \to 0$, in [19, 8, 9].

Under the most basic online model, known as the *adversarial model*, the online algorithm does not know anything about the x_i's or $E(G)$ beforehand. In this model, the seminal result of Karp, Vazirani and Vazirani [16] gives an optimal online $1 - \frac{1}{e}$-competitive algorithm to maximize the size of the matching for *unweighted graphs* where $w_{ij} = 1$ for each $(x_i, y_j) \in E(G)$. For weighted graphs, Mehta et al. [19, 8] presented the first $1 - \frac{1}{e}$-approximation algorithm to maximize the total weight of the allocation for the AdWords problem and this result has been generalized to more general weighted allocation problems [8, 12].

Other than the adversarial model, motivated by applications in online advertising, various *stochastic online* models have been proposed for this problem. In such stochastic models, online nodes $x_i \in X$ arrive in a random order. In other words, given a random permutation $\sigma \in \mathbb{S}_n$, the ball $x_{\sigma(t)}$ arrives at time time t for $t = 1, 2, \ldots, n$. In this case, the seminal result of Devanur and Hayes [9] gives a $1 - \epsilon$-approximation for the problem if the number of balls n is a prior information to the algorithm, and $\frac{\text{OPT}}{w_{ij}} \geq O(\frac{m \log n}{\epsilon^3})$, where $m := |Y|$. This result has been generalized and improved in several followup works [11, 1, 24], and its related models like the iid models with known or unknown distributions have been studied extensively [13, 4, 20, 10][1]. These stochastic models are particularly motivated in the context of online ad allocation. In this context, online nodes correspond to page-views, search queries, or online requests for ads. In these settings, the incoming traffic of page-views may be predicted with a reasonable precision using a vast amount of historical data.

All these stochastic models and their algorithms are useful only if the incoming traffic of online nodes (e.g. page-views) can be predicted with a reasonably good precision. In other words, such algorithms may rely heavily on a precise forecast of the online traffic patterns, and may not react quickly to sudden changes in the traffic. In fact, the slow reaction to such traffic spikes impose a serious limitation in the real-world use of stochastic algorithms in practical applications. This is a common issue in applying stochastic optimization techniques for online resource allocation problems (see e.g., [25]). Various methodologies such as robust or control-based stochastic optimization [5, 6, 25, 23] have been applied to alleviate this drawback. In this paper, we study this problem from a more idealistic perspective and aim to design algorithms that simultaneously achieve optimal approximation ratios for both the adversarial and stochastic models. Our goal is to design algorithms that achieve good performance ratios both in the worst case and in the average case.

Our Contributions and Techniques. In this paper, we study simultaneous approximation algorithms for the adversarial and stochastic models for the online budgeted allocation problem. Our goal is to design algorithms that achieve a competitive ratio strictly better than $1 - 1/e$ on average, while preserving a nearly optimal worst case competitive ratio. Ideally, we want to achieve

[1]In the iid stochastic models, online nodes are drawn iid from a known or an unknown distribution.

the best of both worlds, i.e, to design an algorithm with the optimal competitive ratio in both the adversarial and random arrival models. Toward this goal, we show that this can be achieved for unweighted graphs, but not for weighted graphs. Nevertheless, we present improved approximation algorithms for weighted graphs.

For weighted graphs we prove that no algorithm can simultaneously achieve nearly optimal competitive ratios on both the adversarial and random arrival models. In particular, we show that no online algorithm that achieve an approximation factor of $1 - \frac{1}{e}$ for the worst-case inputs may achieve an average approximation factor better than 97.6% for the random inputs (See Corollary 5.3). More generally, we show that any algorithm achieving an approximation factor of $1 - \epsilon$ in the stochastic model may not achieve a competitive ratio better than $4\sqrt{\epsilon}$ in the adversarial model (See Theorem 5.1). In light of this hardness result, we aim to design algorithms with improved approximation ratios in the random arrival model while preserving the competitive ratio of $1 - \frac{1}{e}$ in the worst case. To this end, we show an almost tight analysis of the algorithm proposed in [19] in the random arrival model. In particular, we show its competitive ratio is at least 0.76, and is no more than 0.81 (See Theorem 3.1, and Lemma 5.5). Combining this with the worst-case ratio analysis of Mehta et al. [19] we obtain an algorithm with the competitive ratio of 0.76 for the random arrival model, while having a $1 - \frac{1}{e}$ ratio in the worst case. It is worth noting that unlike the result of [9] we do not assume any prior knowledge of the number of balls is given to the algorithm.

On the other hand, for unweighted graphs, under some mild assumptions, we show a generalization an algorithm in [14] achieves a competitive ratio of $1 - \epsilon$ in the random arrival model (See Theorem 4.1). Combining this with the worst-case ratio analysis of [14, 19], we obtain an algorithm with the competitive ratio of $1 - \epsilon$ in the random arrival model, while preserving the optimal competitive ratio of $1 - \frac{1}{e}$ in the adversarial model. Previously, a similar result was known for a more restricted stochastic model where all bins have equal capacities [21]. For the case of small degrees, an upper bound of 0.82 is known for the approximation ratio of any algorithm for the online stochastic matching problem (even for the under the iid model with known distributions) [20].

Our proofs consist of three main steps. (i) The main technique is to define an appropriate potential function as an indefinite integral of a scoring function, and interpret the online algorithms as a greedy algorithm acting to improve these potential functions by optimizing the corresponding scoring functions (see Section 2); these potential functions may prove useful elsewhere. (ii) The second important component of the proof is to write a factor-revealing mathematical program based on the potential function and its changes. (iii) Finally, the last part of the proofs involve changing the factor-revealing programs to a constant-size LP and solve it using a solver (in the weighted case), or analyzing the mathematical program explicitly using an intermediary algorithm with an oracle access to the optimum (in the unweighted case). The third step of the proof in the weighted case is inspired by the technique employed by Mahdian and Yan [18] for unweighted graphs, however, the set of mathematical programs we used are quite different from theirs.

All of our results hold under two mild assumptions: (i) large capacities (i.e., $\frac{\max_{i,j} w_{i,j}}{\min_j c_j} \to 0$), and (ii) a mild lower bound on the value of OPT: the aggregate sum of the largest weight ball assigned to each bin by the optimum is much smaller than OPT, i.e., $\sum_j \max_{i:opt(i)=j} w_{i,j} \ll \text{OPT}$. Both of these assumptions are valid in real-world applications of this problem in online advertising. The first assumption also appears in the AdWords problem, and the second assumption aims to get rid of some degenerate cases in which the optimum solution is very small.

Other Related Work. For unweighted graphs, it has been recently observed that the Karp-Vazirani-Vazirani $1-\frac{1}{e}$-competitive algorithm for the adversarial model also achieves an improved approximation ratio of 0.70 in the random arrival model [15, 18]. This holds even without the assumption of large degrees. It is known that without this assumption, one cannot achieve an approximation factor better than 0.82 for this problem (even in the case of iid with known distributions) [20]. This is in contrast with our result for unweighted graphs with large degrees.

Online budgeted allocation and its generalizations appear in two main categories of online advertising: *AdWords (AW)* problem [19, 8, 9], and the *Display Ads Allocation (DA)* problem [12, 11, 1, 24]. In both of these problems, the publisher must assign online page-views (or impressions) to an inventory of ads, optimizing efficiency or revenue of the allocation while respecting pre-specified contracts. In the DA problem, given a set of m advertisers with a set S_j of eligible impressions and demand of at most $n(j)$ impressions, the publisher must allocate a set of n impressions that arrive online. Each impression i has value $w_{ij} \geq 0$ for advertiser j. The goal of the publisher is to assign each impression to one advertiser maximizing the value of all the assigned impressions. The adversarial online DA problem has been studied in [12] in which the authors present a $1-\frac{1}{e}$-competitive algorithm for the DA problem under a free disposal assumption for graphs of large degrees. This result generalizes the $1-\frac{1}{e}$-approximation algorithm by Mehta et al [19] and by Buchbinder et. al. [8]. Following a training-based dual algorithm by Devanur and Hayes [9] for the AW problem, training-based $(1-\epsilon)$-competitive algorithms have been developed for the DA problem and its generalization to packing linear programs [11, 24, 1] including the DA problem. These papers develop a $(1-\epsilon)$-competitive algorithm for online stochastic packing problems in which $\frac{\text{OPT}}{w_{ij}} \geq O(\frac{m\log n}{\epsilon^3})$ (or $\frac{\text{OPT}}{w_{ij}} \geq O(\frac{m\log n}{\epsilon^2})$ applying the technique of [1]) and the demand of each advertiser is large, in the random-order and the i.i.d model. Although studying a similar set of problems, none of the above papers study the simultaneous approximations for adversarial and stochastic models, and the dual-based $1-\epsilon$-competitive algorithms for the stochastic variants do not provide a bounded competitive ratio in the adversarial model.

Dealing with traffic spikes and inaccuracy in forecasting the traffic patterns is a central issue in operations research and stochastic optimization. Various methodologies such as robust or control-based stochastic optimization [5, 6, 25, 23] have been proposed. These techniques either try to deal with a larger family of stochastic models at once [5, 6, 25], try to handle a large class of demand matrices at the same time [25, 2, 3], or aim to design asymptotically optimal algorithms that re-act more adaptively to traffic spikes [23]. These methods have been applied in particular for traffic engineering [25] and inter-domain routing [2, 3]. Although dealing with similar issues, our approach and results are quite different from the approaches taken in these papers. Finally, an interesting related model for combining stochastic and online solutions for the Adwords problem is considered in [17], however their approach does not give an improved approximation algorithm for the iid model.

1.1 Notation

Let $G(X, Y, E)$ be a (weighted) bipartite graph, where $X := \{x_1, \ldots, x_n\}$ is the set of online nodes (or balls), and $Y := \{y_1, \ldots, y_m\}$ is the set of fixed nodes (or bins). For each pair of nodes x_i, y_j, $w_{i,j}$ represents the weight of edge (x_i, y_j). Each online node y_j is associated with a weighted capacity (or budget) $c_j > 0$. The online matching problem is as follows: first a permutation $\sigma \in \mathbb{S}_n$ is chosen (the distribution may be chosen according to any unknown distribution): at times $t = 1, 2, \ldots, n$, the ball $x_{\sigma(t)}$ arrives and its incident edges are revealed to the algorithm. The algorithm can assign this ball to at most one of the bins that are adjacent to it. The total weight of balls assigned to each bin y_j may not exceed its weighted capacity c_j. The objective is to maximize the weight of the final matching.

Given the graph G, the optimum offline solution is the maximum weighted bipartite matching in G respecting the weighted capacities, i.e, the total weight of balls assigned to to a bin y_j may not exceed c_j. For each ball x_i, let $opt(i)$ denote the index of the bin that x_i is being matched to in the optimum solution, and $alg(i)$ be the index of the bin that x_i is matched to in the algorithm. Also for each node $y_j \in Y$, let o_j be the weighted degree of y_j in the optimum solution. Observe that for each j, we have $0 \leq o_j \leq c_j$. By definition, we have the size of the optimum solution is $\mathrm{OPT} = \sum_j o_j$. Throughout the paper ,we use OPT as the total weight of the optimal solution, and ALG as the total weight of the output of the online algorithm.

Throughout this paper, we make the assumption that the weights of the edges are small compared to the capacities, i.e., $\max_{i,j} w_{i,j}$ is small compared to $\min_i c_j$. Also we assume that the aggregate sum of the largest weight ball assigned to each bin by the optimum is much smaller than OPT i.e., $\sum_j \max_{i:opt(i)=j} w_{i,j} \ll \mathrm{OPT}$. In particular, let

$$\gamma \geq \max\left\{ \max_{i,j} \frac{w_{i,j}}{c_j}, \quad \sqrt{\frac{\sum_j \max_{i:opt(i)=j} w_{i,j}}{\mathrm{OPT}}} \right\} \tag{1}$$

the guarantees of our algorithm are provided for the case when $\gamma \to 0$. For justifications behind these mild assumptions, see the discussion in the introduction. Throughout this paper, wlog we assume that the optimum matches all of the balls, otherwise we can throw out the unmatched ball and it can only make the competitive ratio of the algorithm worse.

2 Main Ideas

In this section, we describe the main ideas of the proof. We start by defining the algorithms as deterministic greedy algorithms optimizing specific scoring functions. We also define a concave potential function as an indefinite integral of the scoring function, and show that a "good" greedy algorithm must try to maximize the potential function. In Section 2.1, we show that if σ is chosen uniformly at random, then we can lower-bound the increase of the potential in an ϵ fraction of process; finally in Section 2.2 we write a factor-revealing mathematical program based on the potential function and its changes.

We consider a class of deterministic greedy algorithms that assign each incoming ball $x_{\sigma(t)}$ based on a "scoring function" defined over the bins. Roughly speaking, the scoring function characterizes the "quality" of a bin, and a larger score implies a better-quality bin. In this paper, we assume that the score is independent of the particular labeling of the bins, and it is a non-negative, *non-increasing* function of the amount that is saturated so far (roughly speaking, these algorithms try to prevent over-saturating a bin when the rest are almost empty). Throughout this paper, we assume that the scoring function and its derivative are bounded (i.e., $|f'(.)|, |f(.)| \leq 1$). However, all of our arguments in this section can also be applied to the more general scoring functions that may even depend on the overall capacity c_i of the bins. At a particular time t, let $r_j(t)$ represent the fraction of the capacity of the bin y_j that is saturated so far. Let $f(r_j(t))$ be the score of bin y_j at time t. When the ball $x_{\sigma(t+1)}$ arrives, the greedy algorithm simply computes the score of all of the bins and assigns $x_{\sigma(t+1)}$ to the bin y_j *maximizing the product of* $w_{\sigma(t+1),j}$ *and* $f(r_j(t))$.

Kalyanasundaram, and Pruhs [14] designed the algorithm Balance using the scoring function $f_u(r_j(t)) := 1 - r_j(t)$ (i.e., the algorithm simply assigns an in-coming ball to the neighbor with the smallest ratio if it is less than 1, and drops the ball otherwise). They show that for any unweighted graph G, it achieves a $1 - 1/e$ competitive ratio against any adversarially chosen permutation σ. Mehta et al. [19] generalized this algorithm to weighted graphs by defining the scoring function $f_w(r_j(t)) = (1 - e^{1-r_j(t)})$. Their algorithm, denoted by Weighted-Balance , achieves a competitive ratio of $1 - 1/e$ for the AdWords problem in the adversarial model. We note that both of the algorithms never over-saturate bins (i.e., $0 \leq r_j(t) \leq 1$). Other scoring functions have also been considered for other variants of the problem (see e.g. [17, 12]). Intuitively, these scoring functions are chosen to ensure that the algorithm assigns the balls as close to $opt(x_{\sigma(t)})$ as possible. When the permutation is chosen adversarially, any scoring function would fail to perfectly monitor the optimum assignment (as discussed before, no online algorithm can achieve a competitive ratio better than $1 - 1/e$ in the adversarial model). However, we hope that when σ is chosen uniformly at random, for any adversarially chosen graph G, the algorithm can almost capture the optimum assignment. In the following we try to formalize this observation.

We measure the performance of the algorithm at time t by assigning a potential function that in some sense compares the quality of the overall decisions of the algorithm w.r.t. the optimum. Assuming the optimum solution saturates all of the bins (i.e., $c_j = o_j$), the potential function achieves its maximum at the end of the algorithm if the balls are assigned exactly according to the optimum. The closer the value of the potential function to the optimum means a better assignment of the balls. We define the potential function as the weighted sum of the indefinite integral of the scoring functions of the bins chosen by the algorithm:

$$\phi(t) := \sum_j c_j \int_{r=0}^{r_j(t)} f(r)dr = \sum_j c_j F(r_j(t)).$$

In particular, we use the following potential function for the Balance and the Weighted-Balance algorithms respectively:

$$\phi_u(t) : \quad = \quad -\frac{1}{2}\sum_j c_j(1 - r_j(t))^2 \tag{2}$$

$$\phi_w(t) : \quad = \quad \sum_j c_j(r_j - e^{r_j(t)-1}). \tag{3}$$

Observe that since the scoring function is a non-increasing function of the ratios, its antiderivative $F(.)$ will be a concave function of the ratios. Moreover, since it is always non-negative the value of the potential function never decreases in the running time of the algorithm. By this definition the greedy algorithm can be seen as an online gradient descent algorithm which tries to maximize a concave potential function; for each arriving ball $x_{\sigma(t)}$, it assigns the ball to the bin that makes the largest local increase in the function.

To analyze the performance of the algorithm we lower-bound the increase in the value of the potential function based on the optimum matching. This allows us to show that the final value of the potential function achieved by the algorithm is close to its value in the optimum, thus bound the competitive ratio of the algorithm. In the next section, we use the fact that σ is chosen randomly to lower-bound the increase in ϵn steps. Finally, in section 2.2 we write a factor-revealing mathematical program to compute the competitive ratio of the greedy algorithm.

2.1 Lower bounding the increase in the potential function

In this part, we use the randomness defined on the permutation σ to argue that with high probability the value of the potential function must have a significant increase during the run of the algorithm. We define a particular event $\mathcal{E}$ corresponding to event that the arrival process of the balls is approximately close to its expectation. To show that $\mathcal{E}$ occurs with high probability, we only consider the distribution of arriving balls at $1/\epsilon$ equally distance times; as a result we can mainly monitor the amount of increase in the potential function at these time intervals. For a carefully chosen $0 < \epsilon < 1$, we divide the process into $1/\epsilon$ slabs such that the k^{th} slab includes the $[kn\epsilon + 1, (k+1)n\epsilon]$ balls. Assuming σ is chosen uniformly at random, we show a concentration bound on the weight of the balls arriving in the k^{th} slab. Using that we lower bound $\phi((k+1)n\epsilon) - \phi(kn\epsilon)$ in Lemma 2.2.

First we use the randomness to determine the weight of the balls arriving in the k^{th} slab. Let $I_{i,k}$ be the indicator random variable indicating that the i^{th} ball will arrive in the k^{th} slab. Observe that for any k, the indicators $I_{i,k}$ are *negatively correlated*: knowing that $I_{i,k} = 1$ can only decrease the probability of the occurrence of the other balls in the k^{th} slab (i.e., $\mathbf{P}\left[I_{i,k} | I_{i',k} = 1\right] \leq \mathbf{P}\left[I_{i,k}\right]$). Define $N_{j,k} := \sum_{i:opt(i)=j} w_{i,j} I_{i,k}$ as the sum of the weight of the balls that are matched to the j^{th} bin in the optimum and arrive in the k^{th} slab. It is easy to see that $\mathbf{E}_\sigma\left[N_{j,k}\right] = \epsilon \cdot o_j$, moreover, since it is a linear combination of negatively correlated random variables it will be concentrated around its mean. Define $h(k) := \sum_j |N_{j,k} - \epsilon o_j|$. The following Lemma shows that $h(k)$ is very close to zero for all time slabs k with high probability. Intuitively, this implies that with high probability the balls from each slab are assigned similarly in the optimum solution.

Lemma 2.1. *Let $h(k) := \sum_j |N_{j,k} - \epsilon o_j|$. Then $\mathbf{P}_\sigma\left[\forall k, h(k) \leq \frac{5\gamma}{\epsilon\delta}\mathrm{OPT}\right] \geq 1 - \delta$.*

Proof. It suffices to upper-bound $\mathbf{P}\left[h(k) > \frac{5\gamma}{\epsilon\delta}\mathrm{OPT}\right] \leq \delta\epsilon$; the lemma can then be proved by a simple application of the union bound. First we use Azuma-Hoeffding concentration bound to compute $\mathbf{E}\left[|N_{j,k} - \epsilon o_j|\right]$; then we simply apply the Markov inequality to upper-bound $h(k)$.

Let $W_j := \sqrt{2\sum_{i:opt(i)=j} w_{i,j}^2}$, for any j, k, we show $\mathbf{E}\left[|N_{j,k} - \epsilon o_j|\right] \leq 3W_j$. Since $N_{j,k}$ is a linear combination of negatively correlated random variables $I_{i,k}$ for $opt(i) = j$, and $\mathbf{E}\left[N_{j,k}\right] = \epsilon \cdot o_j$ by a generalization of the Azuma Hoeffding bound to negatively correlated random variables [22] we have

$$\begin{aligned}
\mathbf{E}\left[|N_{j,k} - \epsilon \cdot o_j|\right] &\leq W_j \left\{\sum_{l=0}^{\infty} \mathbf{P}\left[|N_{j,k} - \mathbf{E}\left[N_{j,k}\right]| \geq l \cdot W_j\right]\right\} \leq W_j \left\{1 + 2\sum_{l=1}^{\infty} e^{-\frac{l^2 W_j^2}{2\sum_{i:opt(i)=j} w_{i,opt(i)}^2}}\right\} \\
&\leq W_j \left(1 + 2\sum_{l=1}^{\infty} e^{-l^2}\right) \leq 3W_j. \qquad (4)
\end{aligned}$$

Let $w_{\max}(j) := \max_{i:opt(i)=j} w_{i,j}$. Since W_j^2 as twice the sum of the square of the weights assigned to the j^{th} bin is a convex function we can write $W_j \leq \sqrt{2 w_{\max}(j) o_j}$. Therefore, by the linearity of expectation we have

$$\begin{aligned}\mathbf{E}\left[h(k)\right] &\leq \sum_j 3\sqrt{2w_{\max}(j)o_j} \leq 5\sum_j \frac{w_{\max}(j)/\gamma + \gamma o_j}{2} \\ &\leq 5\{\frac{1}{2\gamma}\sum_j w_{\max}(j) + \frac{\gamma}{2}\mathrm{OPT}\} \leq 5\gamma\mathrm{OPT},\end{aligned}$$

where the last inequality follows from assumption (1). Since $h(k)$ is a non-negative random variable, by the Markov inequality we get $\mathbf{P}\left[h(k) > \frac{5\gamma}{\epsilon\delta}\mathrm{OPT}\right] \leq \delta\epsilon$. The lemma simply follows by applying this inequality for all $k \in \{0, \ldots, 1/\epsilon\}$ and using the union bound. □

Let $\mathcal{E}$ be the event that $\forall k, h(k) \leq \frac{5\gamma}{\epsilon\delta}\mathrm{OPT}$. The next lemma shows that conditioned on $\mathcal{E}$, one can lower-bound the increase in the potential function in any slab (i.e., $\phi((k+1)n\epsilon) - \phi(kn\epsilon)$ for any $0 \leq k < 1/\epsilon$):

Lemma 2.2. *Conditioned on $\mathcal{E}$, for any $0 \leq k < 1/\epsilon$, $t_0 = kn\epsilon$, and $t_1 = (k+1)n\epsilon$ we have*

$$\phi(t_1) - \phi(t_0) \geq \epsilon\sum_j \{o_j f(r_j(t_1))\} - \frac{6\sqrt{\gamma}}{\epsilon\delta}\mathrm{OPT}.$$

Proof. First we simply compute the increase of the potential function at time $t+1$, for some $t_0 \leq t < t_1$. Then, we lower-bound the increase using the monotonicity of the scoring function $f(.)$. Finally, we condition on $\mathcal{E}$ and lower-bound the final expression in terms of OPT. Let $\sigma(t+1) = i$, and assume the algorithm assigns x_i to the j^{th} bin (i.e., $alg(i) = j$); since the algorithm maximizes $w_{i,j}f(r_j(t))$ we can lower-bound the increase of the potential function based on the optimum. First using the mean value theorem of the calculus we have:

$$c_j\left\{F(r_j(t) + \frac{w_{i,j}}{c_j}) - F(r_j(t)\right\} = c_j\left\{\frac{w_{i,j}}{c_j}f(r_j(t)) + \frac{1}{2}\left(\frac{w_{i,j}}{c_j}\right)^2 f'(r^*)\right\},$$

for some $r^* \in [r_j(t), r_j(t) + w_{i,j}/c_j]$. Since the derivative of $f(.)$ is bounded (i.e., $|f'(r)| \leq 1$ for all $r \in [0,1]$), we get

$$\begin{aligned}\phi(t+1) - \phi(t) &= c_j\left\{F(r_j(t) + \frac{w_{i,j}}{c_j}) - F(r_j(t)\right\} \geq w_{i,opt(i)}f(r_{opt(i)}(t)) - w_{i,j}\frac{w_{i,j}}{c_j} \\ &\geq w_{i,opt(i)}f(r_{opt(i)}(t_1)) - \gamma w_{i,j}, \qquad (5)\end{aligned}$$

where the first inequality follows by the greedy decision chosen by the algorithm $w_{i,j}f(r_j(t)) \geq w_{i,opt(i)}f(r_{opt(i)}(t))$, and the last inequality follows by assumption (1).

Consider a single run of the algorithm; wlog we assume that ALG $\leq$ OPT. We can monitor the amount of increase in the potential function in the k^{th} slab as follows:

$$\begin{aligned}\phi(t_1) - \phi(t_0) &\geq \sum_{t=t_0}^{t_1-1}\left\{w_{\sigma(t),opt(\sigma(t))}f(r_{opt(\sigma(t))}(t)) - \gamma w_{\sigma(t),alg(\sigma(t))}\right\} \\ &\geq \sum_j \sum_{t_0 \leq t < t_1, opt(\sigma(t))=j} f(r_j(t_1))w_{\sigma(t),j} - \gamma\mathrm{OPT} \\ &= \sum_j f(r_j(t_1))N_{j,k} - \gamma\mathrm{OPT}\end{aligned}$$

where the second inequality follows by the fact that $f(.)$ is a non-increasing function of the ratio, and $\sum_{t_0 \le t < t_1} w_{\sigma(t),alg(\sigma(t))} \le \text{ALG} \le \text{OPT}$, and the equality follows from the definition of $N_{j,k}$. By lemma 2.1 we know $N_{j,k}$ is highly concentrated around $\epsilon \cdot o_j$. Conditioned on $\mathcal{E}$, we have $h(k) \le \frac{5\gamma}{\epsilon\delta} OPT$, thus:

$$\begin{aligned}
\phi(t_1) - \phi(t_0) &\ge \epsilon \sum_j f(r_j(t_1)) o_j - \sum_j |N_{j,k} - \epsilon o_j| - \gamma \text{OPT} \ge \epsilon \sum_j f(r_j(t_1)) o_j - h(k) - \gamma \text{OPT} \\
&\ge \epsilon \sum_j f(r_j(t_1)) o_j - \frac{6\gamma}{\epsilon\delta} \text{OPT}
\end{aligned}$$

where the first inequality follows by the assumption $|f(.)| \le 1$.

□

2.2 Description of the factor-revealing Mathematical Program

In this section we propose a factor-revealing mathematical program that lower-bounds the competitive ratio of the algorithms Balance and Weighted-Balance . In Sections 3, and 4 we derive a relaxation of the program and analyze that relaxation. Interestingly, the main non-trivial constraints follows from the lower-bounds obtained for the amount of increase in the potential function in Lemma 2.2.

The details of the program is described in MP(1). It is worth noting that the last inequality in this program follows from the monotonicity property of the ratios. In other words, we assume the ratio function $r_j(t)$ is a monotonically increasing function w.r.t. to t.

$$\begin{array}{llcll}
\text{MP(1)} & \text{minimize } \frac{1}{\text{OPT}} \sum_j \min\{r_j(n), 1\} c_j & & & \\
\text{s.t.} & \sum_j c_j F(r_j(t)) & = & \phi(t) & \forall t \in [n], \\
 & \epsilon \sum_j o_j f(r_j((k+1)n\epsilon)) - \frac{6\gamma}{\epsilon\delta} \text{OPT} & \le & \phi((k+1)n\epsilon) - \phi(kn\epsilon) & \forall k \in [\frac{1}{\epsilon} - 1], \\
 & o_j & \le & c_j & \forall j \in [m], \\
 & \sum_j o_j & = & \text{OPT}, & \\
 & r_j(t) & \le & r_j(t+1) & \forall j, t \in [n-1].
\end{array}$$

The following proposition summarizes our arguments and shows that the program MP(1) is a relaxation for any deterministic greedy algorithm that works based on a scoring function. It is worth noting that the whole argument still follows when the scoring function is not necessarily non-negative; we state the proposition in this general form.

Proposition 2.3. *Let f be any non-increasing, scoring function of the ratios $r_j(t)$ of the bins such that $|f(r)|, |f'(r)| \le 1$ for the range of ratios that may be encountered in the running time of the algorithm. For any (weighted) graph $G = (X, Y)$, and $\epsilon > 0$, with probability at least $1 - \delta$,* MP(1) *is a factor-revealing mathematical program for the greedy deterministic algorithm that uses scoring function $f(.)$.*

Since the potential function $F(.)$ is a concave function, this program may not be solvable in polynomial time. In section 4, we show that after adding a new constraint it is possible to analyze it analytically for the unweighted graphs. To deal with this issue for the weighted graphs, we write a constant-size LP relaxation of the program that lower-bounds the optimum solution (after losing a small error). Finally, we solve the constant-size LP by an LP solver, and thus obtain a nearly tight bound for the competitive ratio of the Weighted-Balance (see section 3 for more details).

In the rest of this section, we write a simpler mathematical program that will be used later in section 3 for analyzing the Weighted-Balance algorithm. In particular, we simplify the critical constraint that measures the increase in the potential function by further removing the term $-\frac{6\gamma}{\epsilon\delta}\mathrm{OPT}$. Moreover, since Weighted-Balance never over-saturates bins we can also add the constraint $r_j(n) \le 1$ to both MP(1) and MP(2) and still have a relaxation of Weighted-Balance .

$$\begin{array}{llcll}
\mathrm{MP}(2) & \text{minimize } \sum_j r_j(n)c_j & & & \\
\text{s.t.} & \sum_j c_j(r_j(t) - e^{r_j(t)-1}) & = & \phi(t) & \forall t \in [n], \\
 & \epsilon \sum_j o_j(1 - e^{r_j((k+1)n\epsilon)-1}) & \le & \phi((k+1)n\epsilon) - \phi(kn\epsilon) & \forall k \in [\frac{1}{\epsilon}], \\
 & o_j & \le & c_j & \forall j \in [m], \\
 & \sum_j o_j & = & 1. & \\
 & r_j(t) & \le & r_j(t+1) & \forall j, t \in [n-1], \\
 & r_j(n) & \le & 1 & \forall j \in [m].
\end{array}$$

In the next Lemma we show that the optimum value of MP(2) is at least $(1 - \sqrt{\frac{12\gamma}{\epsilon^2\delta}})$ of MP(1):

Lemma 2.4. *For any weighted graph G, we have* $\mathrm{MP}(1) \ge (1-\alpha)\min\{1, \mathrm{MP}(2)\}$, *where* $\alpha := \sqrt{\frac{12\gamma}{\epsilon^2\delta}}$.

Proof. Wlog we can replace OPT $= 1$ in MP(1). Let $s_1 := \{r_j(t), c_j, o_j, \phi(t)\}$ be a feasible solution of the MP(1). If $\sum_j r_j(n)c_j \ge (1-\alpha)$ we are done; otherwise we construct a feasible solution s_2 of the MP(2) such that the value of s_1 is at least $(1-\alpha)$ of the value of s_2. Then the lemma simply follows from the fact that the cost of the value of the optimum solution of MP(1) is at least of $(1-\alpha)$ of the value of the optimum of MP(2).

Define $s_2 := \{r_j(t), c_j/(1-\alpha), o_j, \phi(t)/(1-\alpha)\}$. Trivially, s_2 satisfies all except (possibly) the second constraint of MP(2). Moreover, the value of s_1 is $(1-\alpha)$ times the value of s_2. It remains to prove the feasibility of the second constraint of MP(2), i.e.,

$$\epsilon(1-\alpha)\sum_j o_j(1 - e^{r_j((k+1)n\epsilon)-1}) \le \phi((k+1)n\epsilon) - \phi(kn\epsilon),$$

for all $k \in [1/\epsilon]$. Since s_1 is a feasible solution of MP(1) we have

$$\begin{aligned}
\phi((k+1)n\epsilon) - \phi(kn\epsilon) &\ge \epsilon\sum_j o_j(1 - e^{r_j((k+1)n\epsilon)-1}) - \frac{\epsilon}{2}\alpha^2 \\
&\ge \epsilon\sum_j o_j(1 - e^{r_j((k+1)n\epsilon)-1})\left\{1 - \frac{\frac{\epsilon}{2}\alpha^2}{\frac{\epsilon}{2}\sum_j o_j(1 - r_j((k+1)n\epsilon)}\right\}, \qquad (6)
\end{aligned}$$

where the last inequality follows from the assumption that $0 \le r_j(t) \le 1$, and the fact that $1 - e^{x-1} \ge \frac{1}{2}(1-x)$ for $x \in [0,1]$. On the other hand, since $\sum_j r_j(n)c_j < 1-\alpha$, we can write:

$$\sum_j o_j(1 - r_j((k+1)n\epsilon)) \ge 1 - \sum_j c_j r_j(n) \ge \alpha$$

The lemma simply follows from putting the above inequality together with equation (6). □

3 The Competitive Ratio of Weighted-Balance

In this section, we lower-bound the competitive ratio of the WEIGHED-BALANCE algorithm in the random arrival model. More specifically, we prove the following theorem:

Theorem 3.1. *For any weighted graph $G = (X, Y)$, and $\delta > 0$, with probability $1-\delta$, the competitive ratio of the* Weighted-Balance *algorithm in the random arrival model is at least $0.76(1 - O(\sqrt{\gamma/\delta}))$.*

To prove the bound in this theorem, we write a constant-size linear programming relaxation of the problem based on MP(2) and solve the problem by an LP solver. The main two difficulties with solving program MP(2) are as follows: first, as we discussed in section 2.2, MP(2) is not a convex program; second, the size of the program (i.e., the number of variables and constraints) is a function of the size of the graph G. The main idea follows from a simple observation that the main inequalities in MP(2), those lower-bounding the increase in the potential function, are indeed lower-bounding the increase in the potential function only at constant (i.e., $1/\epsilon$) number of times. Hence, we do not need to keep track of the ratios and the potential function for all $t \in [n]$; instead it suffices to monitor these values at $1/\epsilon$ critical times (i.e., at times $kn\epsilon$ for $k \in [1/\epsilon]$), for a constant ϵ. Even in those critical times it suffices to approximately monitor the ratios of the bins by discretizing the ratios into $1/\epsilon$ slabs.

For any integers $0 \le i < 1/\epsilon, 0 \le k \le 1/\epsilon$, let $c_{i,k}$ be the sum of the capacities of the bins of ratio $r_j(kn\epsilon) \in [i\epsilon, (i+1)\epsilon)$, and $o_{i,k}$ be the sum of the weighted degree of the bins of ratio $r_j(kn\epsilon) \in [i\epsilon, (i+1)\epsilon)$ in the optimum solution, i.e.,

$$c_{i,k} := \sum_{j : r_j(kn\epsilon) \in [i\epsilon, (i+1)\epsilon)} c_j, \qquad o_{i,k} := \sum_{j : r_j(kn\epsilon) \in [i\epsilon, (i+1)\epsilon)} o_j. \tag{7}$$

Now we are ready to describe the constant-size LP relaxation of MP(2). We write the LP relaxation in terms of the new variables $c_{i,k}, o_{i,k}$. In particular, instead of writing the constraints in terms of the actual ratios of the bins, we round down (or round up) the ratios to the nearest multiple of ϵ such that the constraint remains satisfied. The details are described in LP(1).

$$\begin{array}{llcll}
\text{LP}(1) & \text{minimize } \frac{1}{1-1/e}\left\{\phi(\frac{1}{\epsilon}) - \sum_{i=0}^{1/\epsilon-1} c_{i,k}(i\epsilon/e - e^{i\epsilon-1})\right\} & & & \\
\text{s.t.} & \sum_{i=0}^{1/\epsilon-1} c_{i,k}(i\epsilon - e^{i\epsilon-1}) & \le & \phi(k) & \forall k \in [\frac{1}{\epsilon}] \\
& \sum_{i=0}^{1/\epsilon-1} \epsilon o_{i,k+1}(1 - e^{(i+1)\epsilon-1}) & \ge & \phi(k+1) - \phi(k) & \forall k \in [\frac{1}{\epsilon} - 1] \\
& c_{i,k} & \ge & o_{i,k} & \forall i \in [\frac{1}{\epsilon} - 1], k \in [\frac{1}{\epsilon}] \\
& \sum_{i=0}^{1/\epsilon-1} o_{i,k} & = & 1 & \forall k \in [\frac{1}{\epsilon}] : \\
& \sum_{l=i}^{1/\epsilon-1} c_{l,k+1} & \ge & \sum_{l=i}^{1/\epsilon-1} c_{l,k} & \forall i \in [\frac{1}{\epsilon} - 1], k \in [\frac{1}{\epsilon} - 1]
\end{array}$$

In the next Lemma, we show that the LP(1) is a linear programming relaxation of the program MP(2):

Lemma 3.2. *The optimum value of* LP(1) *lower-bounds the optimum value of* MP(2).

Proof. We show that for any feasible solution $s := \{r_j(t), c_j, o_j, \phi(t)\}$ of MP(2) we can construct a feasible solution $s' = \{c'_{i,k}, o'_{i,k}, \phi'(k)\}$ for LP(1) with a smaller objective value. In particular, we construct s' simply by using equation (7), and letting $\phi'(k) := \phi(kn\epsilon)$. First we show that all constraints of LP(1) are satisfied by s', then we show that the value of LP(1) for s' is smaller than the value of MP(2) for s.

The first equation of LP(1) simply follows from rounding down the ratios in the first equation of MP(2) to the nearest multiple of ϵ. The equation remains satisfied by the fact that the potential function $\phi(.)$ is increasing in the ratios (i.e., $F_w(r) = r - e^{r-1}$ is increasing in $r \in [0,1]$). Similarly, the second equation of LP(1) follows from rounding up the ratios in the second equation of MP(2), and noting that the scoring function is decreasing in the ratios (i.e., $f_w(r) = 1 - e^{r-1}$ is decreasing for $r \in [0,1]$). The third and fourth equations can be derived from the corresponding equations in MP(2). Finally, the last equation follows from the monotonicity property of the ratios (i.e., $r_j(t)$ is a non-decreasing function of t).

It remains to compare the values of the two solutions s, and s'. We have

$$\begin{aligned}\frac{1}{1-1/e}\left\{\phi'(\frac{1}{\epsilon}) - \sum_{i=0}^{1/\epsilon-1} c'_{i,1/\epsilon}\left(i\epsilon/e - e^{i\epsilon-1}\right)\right\} &\leq \frac{1}{1-1/e}\left\{\phi(n) - \sum_j c_j(\frac{r_j(n)}{e} - e^{r_j(n)-1})\right\} \\ &= \sum_j c_j r_j(n),\end{aligned}$$

where the inequality follows from the fact that $r/e - e^{r-1}$ is a decreasing function for $r \in [0,1]$, and the last inequality simply follows from the definition of $\phi_w(.)$ (i.e., the first equation of MP(2)). $\square$

Now we are ready to prove Theorem 3.1:
Proof of Theorem 3.1. By Proposition 2.3, for any $\epsilon > 0$, with probability $1-\delta$ the competitive ratio of Weighted-Balance is lower-bounded by the optimum of MP(1). On the other hand, by Lemma 2.4 the optimum solution of MP(1) is at least $(1 - \sqrt{\frac{12\gamma}{\epsilon^2\delta}})$ of the optimum solution of MP(2). Finally, by Lemma 3.2 the optimum solution of MP(2) is at least the optimum of LP(1). Hence, with probability $1-\delta$ the competitive ratio of Weighted-Balance is at least $(1 - \sqrt{\frac{12\gamma}{\epsilon^2\delta}})$ of the optimum of LP(1).

The constant-size linear program LP(1) can be solved numerically for any value of $\epsilon > 0$. By solving this LP using an LP solver, we can show that for $\epsilon = 1/250$ the optimum solution is greater than 0.76. This implies that with probability $1-\delta$ the competitive ratio of Weighted-Balance is at least $0.76(1 - O(\sqrt{\gamma/\delta}))$. $\square$

Remark 3.3. *We also would like to remark that the optimum solution of* LP(1) *beats the* $1 - 1/e$ *factor even for* $\epsilon = 1/8$*; roughly speaking this implies that even if the permutation* σ *is almost random, in the sense that each* $1/8$ *of the input almost has the same distribution, then* Weighted-Balance *beats the* $1 - 1/e$ *factor.*

4 The Competitive Ratio of Balance

In this section we show that for any unweighted graph G, under some mild assumptions, the competitive ratio of Balance approaches 1 in the random arrival model.

Theorem 4.1. *For any unweighted bipartite graph* $G = (X, Y, E)$*, and* $\delta > 0$*, with probability* $1 - \delta$ *the competitive ratio of* Balance *in the random arrival model is at least* $1 - \frac{\beta \sum_i c_j}{\text{OPT}}$*, where* $\beta := 3(\gamma/\delta)^{1/6}$.

First we assume that our instance is *all-saturated* meaning that the optimum solution saturates all of the bins (i.e., $c_j = o_j$ for all j), and show that the competitive ratio of the algorithm is at least $1 - 3(\gamma/\delta)^{1/6}$:

Lemma 4.2. *For any $\delta > 0$, with probability $1-\delta$ the competitive ratio of* Balance *on all-saturated instances in the random arrival model is at least $1 - \beta$.*

Then we prove Theorem 4.1 via a simple reduction to the all-saturated case.

To prove Lemma 4.2, we analyze a slightly different algorithm Balance' that always assigns an arriving ball (possibly to an over-saturated bin); this will allow us to keep track of the number of assigned balls at each step of the process. In particular we have

$$\forall t \in [n] : \sum_j c_j r_j(t) = t, \tag{8}$$

where $r_j(t)$ does not necessarily belong to $[0, 1]$. The latter certainly violates some of our assumptions in Section 2. To avoid the violation, we provide some additional knowledge of the optimum solution to Balance' such that the required assumptions are satisfied, and it achieves exactly the same weight as Balance .

We start by describing Balance' , then we show that it still is a feasible algorithm for the potential function framework studied in Section 2; in particular we show it satisfies Proposition 2.3. When a ball x_i arrives at time $t+1$ (i.e., $\sigma(t+1) = i$), similar to Balance , Balance' assigns it to a bin maximizing $w_{i,j} f_u(r_j(t))$; let j be such a bin. Unlike Balance if $r_j(t) \geq 1$ (i.e., all neighbors of x_i are saturated), we do not drop x_i; instead Balance' assigns it to the bin $y_{opt(i)}$.

First note that although Balance' magically knows the optimum assignment of a ball once all of its neighbors are saturated, it achieves the same weight matching. This simply follows from the fact that over-saturating bins does not increase our gain, and does not alter any future decisions of the algorithm. Next we use Proposition 2.3 to show that MP(1) is indeed a mathematical programming relaxation for Balance' .

By Proposition 2.3, we just need to verify that $|f_u(.)|, |f'_u(.)| \leq 1$ for all the ratios we might encounter in the run of Balance' . Since $f_u(r) = (1 - r)$, and the ratios are always non-negative, it is sufficient to show that the ratios are always upper-bounded by 2. To prove this, we crucially use the fact that Balance' has access to the optimum assignment for the balls assigned to the over-saturated bins. Observe that the set of balls assigned to a bin after it is being saturated, is always a subset of the balls assigned to it in the optimum assignment. Since the ratio of all bins are at most 1 in the optimum, they will be upper-bounded by 2 in Balance' .

The following is a simple mathematical programming relaxation to analyze Balance' in the all-saturated instances:

$$\begin{array}{llcll}
\text{MP}(3) & \text{minimize } \sum_j \min\{r_j(n), 1\} c_j & & & \\
\text{s.t.} & \sum_j c_j r_j(t) & = & t & t \in [n], \\
 & \epsilon \sum_j c_j(1 - r_j((k+1)n\epsilon)) - \frac{6\gamma}{\epsilon\delta}\text{OPT} & \leq & \phi_u((k+1)n\epsilon) - \phi_u(kn\epsilon) & \forall k \in [\frac{1}{\epsilon} - 1],
\end{array}$$

Note that the first constraint follows from (8), and the second constraint follows from the second constraint of MP(1), and the fact that $c_j = o_j$ in the all-saturated instances.

Now we are ready to prove Lemma 4.2:

Proof of Lemma 4.2. With probability $1 - \delta$, MP(3) is a mathematical programming relaxation of Balance' . First we sum up all $1/\epsilon$ second constraints of MP(3) to obtain a lower-bound on $\phi_u(n)$, and we get $\phi_u(n)$ is very close to zero (intuitively, the algorithm almost manages to optimize the potential function). Then, we simply apply the Cauchy-Schwarz inequality to $\phi_u(n)$ to bound the loss of Balance' .

We sum up the second constraint of MP(3) for all $k \in \{0, 1, \dots, \frac{1}{\epsilon} - 1\}$; the RHS telescopes and we obtain:

$$\begin{aligned}\phi_u(n) - \phi_u(0) &\geq OPT(1 - \frac{6\gamma}{\epsilon^2\delta}) - \epsilon \sum_{k=0}^{1/\epsilon-1} \sum_j c_j r_j((k+1)n\epsilon) \geq n(1 - \frac{6\gamma}{\epsilon^2\delta}) - \epsilon^2 n \sum_{k=0}^{1/\epsilon-1}(k+1) \\ &\geq n(\frac{1}{2} - \frac{\epsilon}{2} - \frac{6\gamma}{\epsilon^2\delta})\end{aligned}$$

where the first inequality follows by the assumption that the instance is all-saturated, and the second inequality follows from applying the first constraint of MP(3) for $t = (k+1)n\epsilon$, and the fact that $\mathrm{OPT} = n$. Since $\phi_u(0) = -\frac{1}{2}\sum_j (1 - r_j(0))^2 = -n/2$, we obtain $\phi_u(n) \geq -n(\frac{\epsilon}{2} + \frac{6\gamma}{\epsilon^2\delta})$.

Observe that only the non-saturated bins incur a loss to the algorithm, i.e.,

$$Loss(\text{Balance' }) = \sum_{r_j(n)<1} c_j(1 - r_j(n)).$$

Using the lower-bound on $\phi_u(n)$ we have

$$\sum_{r_j(n)<1} c_j(1 - r_j(n)) \leq \sqrt{\sum_{r_j(n)<1} c_j(1 - r_j(n))^2 \cdot \sum_{r_j(n)<1} c_j} \leq \sqrt{-2\phi_u(n)\cdot n} \leq n\sqrt{\epsilon + \frac{12\gamma}{\epsilon^2\delta}},$$

where the first inequality follows by the Cauchy-Schwarz inequality, and the second inequality follows from the definition of $\phi_u(n)$. The lemma simply follows from choosing $\epsilon = 2(2\gamma/\delta)^{1/3}$ in the above inequality. $\square$

Next we prove Theorem 4.1; we analyze the general instances by a reduction to all-saturated instances.

Proof of Theorem 4.1. Let $G = (X, Y)$ be an unweighted graph, similar to Lemma 4.2 it is sufficient to analyze Balance' on G. For every bin y_j we introduce $c_j - o_j$ dummy balls that are only adjacent to the j^{th} bin, and let $G' = (X', Y)$ be the new instance. First we show that the expected number of non-dummy balls matched by Balance' in G' is at most the expected size of the matching that Balance' achieves in G. We analyze the performance of Balance' on G simply using Lemma 4.2, and eliminating the effect of dummies.

Fix a permutation $\sigma \in \mathbb{S}_{|X'|}$; let $W'(\sigma)$ be the number of non-dummy balls matched by Balance' on σ. Similarly, let $W(\sigma[X])$ be the size of the matching obtained on $\sigma[X]$ in G, where $\sigma[X]$ is the projection of σ on X. Using an argument similar to [7, Lemma 2] (e.g., the monotonicity property), one can show that $W'(\sigma) \leq W(\sigma[X])$ for all $\sigma \in \mathbb{S}_{|X'|}$. Hence, to compute the competitive ratio of Balance' on G, it is sufficient to upper-bound the expected number of non-dummy balls not-matched by Balance' on G'. The latter is certainly not more than the total loss of Balance' on G' which is no more than $\beta \sum_j c_j$ by Lemma 4.2. $\square$

5 Hardness Results

In this section, we show that there exists a family of weighted graphs G such that for any $\epsilon > 0$, any online algorithm that achieves a $1 - \epsilon$ competitive ratio in the random arrival model, does not achieve an approximation ratio better than a function $g(\epsilon)$ in the adversarial model, where $g(\epsilon) \to 0$ as $\epsilon \to 0$. More specifically, we prove something stronger:

Theorem 5.1. *For any constants $\delta, \epsilon > 0$, there exists family of weighted bipartite graphs $G = (X, Y)$ such that for any (randomized) algorithm that achieves $1-\epsilon$ competitive ratio (in expectation) on at least δ fraction of the permutations $\sigma \in \mathbb{S}_{|X|}$, does not achieve more than $4\sqrt{\epsilon}$ (in expectation) for a particularly chosen permutation in another graph G'.*

As a corollary, we can show that any algorithm that achieves the competitive ratio of $1 - 1/e$ in the adversarial model can not achieve an approximation factor better than 0.976 in the random arrival model. Moreover, at the end of this section, we show that for some family of graphs the Weighted-Balance algorithm does not achieve an approximation factor better than 0.81 in the random arrival model (see Lemma 5.5 for more details). This implies that our analysis of the competitive ratio of this algorithm is tight up to an additive factor of 5%. We start by presenting the construction of the hard examples:

Example 5.2. *Fix a large enough integer $l > 0$, and let $\alpha := \sqrt{\epsilon}$; let $Y := \{y_1, y_2\}$ with capacities $c_1 = c_2 = l$. Let C and D be two types of balls (or online nodes), and let the set of online nodes X correspond to a set of l copies of C and l/α copies of D. Each type C ball has a weight of 1 for y_1, and a weight of 0 for y_2, while a type D ball has a weight of 1 in y_1 and a weight of α in y_2.*

First of all, observe that the optimum solution achieves a matching of weight $2l$ simply by assigning all type C balls to y_1, and type D balls to y_2. On the other hand, any algorithm that achieves the competitive ratio of $1-\epsilon$ in the random arrival model should match the balls in a way "very similar" to this strategy. However, if the algorithm uses this strategy, then an adversary may construct an instance by preserving the first l balls of the input followed by l/α dummy balls. But in this new instance it is "much better" to assign all of the first l balls to y_1. In the following we formalize this observation.

Proof of Theorem 5.1. Let G be the graph constructed in Example 5.2, and let A be a (randomized) algorithm that achieves $1-\epsilon$ competitive ratio (in expectation) on at least δ fraction of permutations $\sigma \in \mathbb{S}_n$, where $n = l + l/\alpha$, for some constant $\delta > 0$. First we show that there exists a particular permutation σ^* such that there are at most $l\alpha$ balls of type C among $\{\sigma^*(1), \ldots, \sigma^*(l)\}$, and algorithm A achieves at least $(1-\epsilon)2l$ on σ^*. Then we show that the (expected) gain of A from the first l balls is at most $4l\sqrt{\epsilon}$. Finally, we construct a new graph $G' = (X', Y)$ and a permutation σ' such that the first l balls in σ' is the same as the first l balls of σ^*. This will imply that A does not achieve a competitive ratio better than $4\sqrt{\epsilon}$ on G'.

To find σ^* it is sufficient to show that with probability strictly more than $1-\delta$ the number of type A balls among the first l balls of a uniformly random chosen permutation σ is at most $l\alpha$. This can be proved simply using the Chernoff-Hoeffding bound. Let B_i be a Bernoulli random variable indicating that $x_{\sigma(i)}$ is of type C, for $1 \le i \le l$. Observe that $\mathbf{E}_\sigma[B_i] = \frac{\alpha}{1+\alpha}$, and these variables are negatively correlated. By a generalization of Chernoff-Hoeffding bound [22] we have

$$\mathbf{P}\left[\sum_{i=1}^{l} B_i > \alpha l\right] \le e^{-\frac{l\alpha^3}{6}} < \delta,$$

where the last inequality follows by choosing l large enough. Hence, there exists a permutation σ^* such that the number of type C balls among its first l balls is at most $l\alpha$, and A achieves $(1-\epsilon)2l$ on σ^*.

Next we show that the (expected) gain of A from the first l balls of σ^* is at most $2l(\alpha + \epsilon/\alpha) = 4l\sqrt{\epsilon}$. This simply follows from the observation that any ball of type D that is assigned to y_1 incurs a loss of α. Since the expected loss of the algorithm is at most $2l\epsilon$ on σ^*, the expected number of type D balls assigned to y_1 (in the whole process) is no more than $\frac{2l\epsilon}{\alpha}$. We can upper-bound the (expected) gain of the algorithm from the first l balls by $l\alpha + \frac{2l\epsilon}{\alpha} + l\alpha$, where the first term follows from the upper-bound on the number of A balls, and the last term follows from the number of B balls (that may possibly be) assigned to y_2.

It remains to construct the adversarial instance G' together with the permutation σ'. G' has the same set of bins, while X' is the union of the first l balls of σ^* with l/α dummy balls (a dummy ball has zero weight in both of the bins). We construct σ' by preserving the first l balls of σ^*, filling the rest with the dummy balls (i.e., $x_{\sigma'(i)} = x_{\sigma^*(i)}$ for $1 \le i \le l$). First, observe that the optimum solution in G' achieves a matching of weight l simply by assigning all of the first l balls to y_1. On the other hand, as we proved the (expected) gain of the algorithm A is no more than $4l\sqrt{\epsilon}$ on G'. Therefore, the competitive ratio of A in this adversarial instance is no more than $4\sqrt{\epsilon}$. □

The following corollary can be proved simply by choosing δ small enough in Theorem 5.1:

Corollary 5.3. *For any constant $\epsilon > 0$, any algorithm that achieves a competitive ratio of $1-\epsilon$ in the random arrival model does not achieve strictly better than $4\sqrt{\epsilon}$ in the adversarial model. In particular, it implies that any algorithm that achieves a competitive ratio of $1-\frac{1}{e}$ in the adversarial model does not achieve strictly better than 0.976 in the random order model.*

It is also worth noting that Weighted-Balance achieves at least 0.89 competitive ratio in the random arrival model for Example 5.2, and the worst case happens for $\alpha \approx 0.48$. Next we present a family of examples where the Weighted-Balance does not achieve a factor better than 0.81 in the random arrival model.

Example 5.4. *Fix a large enough integer $n > 0$, and $\alpha < 1$; again let $Y := \{y_1, y_2\}$ with capacities $c_l = n$, and $c_2 = n^2$. Let X be a union of n identical balls each of weight 1 for y_1 and α for y_2.*

Lemma 5.5. *For a sufficiently large n, and a particularly chosen $\alpha > 0$, the competitive ratio of the* Weighted-Balance *in the random arrival model for Example 5.4 is no more than* 0.81.

Proof. First observe that the optimum solution achieves a matching of weight n simply by assigning all balls to y_1. Intuitively, Weighted-Balance starts with the same strategy, but after partially saturating y_1, it sends the rest to y_2 (note that each ball that is sent to y_2 incurs a loss of $1-\alpha$ to the algorithm). Recall that $r_1(n)$ is the ratio of y_1 at the end of the algorithm. The lemma essentially follows from upper-bounding $r_1(n)$ by $1+1/n+\ln(1-\alpha(1-e^{1/n-1}))$. Since the algorithm achieves a matching of weight exactly $r_1(n)n + (1-r_1(n))n\alpha$, and $\mathrm{OPT} = n$, the competitive ratio is $r_1(n) + (1-r_1(n))\alpha$. By optimizing over α, one can show that the minimum competitive ratio is no more than 0.81, and it is achieved by choosing $\alpha \simeq 0.55$.

It remains to show that $r_1(n) \le 1 + 1/n + \ln(1-\alpha(1-e^{1/n-1}))$. Let t be the last time where a ball is assigned to y_1 (i.e., $r_1(t-1) + 1/n = r_1(t) = r_1(n)$). Since the ball at time t is assigned to y_1, we have

$$1 \cdot f_w(r_1(t-1)) \ge \alpha \cdot f_w(r_2(t-1)) \ge \alpha \cdot f_w(\frac{1}{n}),$$

where the last inequality follows by the fact that the ratio of the second bin can not be more than $\alpha \cdot n/c_2 < 1/n$, and $f_w(.)$ is a non-increasing function of the ratios. Using $f_w(r) = 1 - e^{r-1}$, and $r_1(t-1) + 1/n = r_1(n)$ we obtain that $r_1(n) \le 1 + 1/n + \ln(1-\alpha(1-e^{1/n-1}))$. □

References

[1] S. Agrawal, Z. Wang, and Y. Ye. A dynamic near-optimal algorithm for online linear programming. *Computing Research Repository*, 2009. 2, 4

[2] D. Applegate and E. Cohen. Making routing robust to changing traffic demands: algorithms and evaluation. *IEEE/ACM Trans. Netw.*, 16(6):1193–1206, 2006. 4

[3] Y. Azar, E. Cohen, A. Fiat, H. Kaplan, and H. Räcke. Optimal oblivious routing in polynomial time. *J. Comput. Syst. Sci.*, 69(3):383–394, 2004. 4

[4] B. Bahmani and M. Kapralov. Improved bounds for online stochastic matching. In *ESA*, pages 170–181, 2010. 2

[5] A. Ben-Tal and A. Nemirovski. Robust optimization - methodology and applications. *Math. Program.*, 92(3):453–480, 2002. 2, 4

[6] D. Bertsimas, D. Pachamanova, and M. Sim. Robust linear optimization under general norms. *Oper. Res. Lett.*, 32(6):510–516, 2004. 2, 4

[7] B. Birnbaum and C. Mathieu. On-line bipartite matching made simple. *SIGACT News*, 39:80–87, March 2008. 14

[8] N. Buchbinder, K. Jain, and J. Naor. Online Primal-Dual Algorithms for Maximizing Ad-Auctions Revenue. In *Proc. ESA*, page 253. Springer, 2007. 2, 4

[9] N. Devanur and T. Hayes. The adwords problem: Online keyword matching with budgeted bidders under random permutations. In *ACM EC*, 2009. 1, 2, 3, 4

[10] N. R. Devanur, K. Jain, B. Sivan, and C. A. Wilkens. Near optimal online algorithms and fast approximation algorithms for resource allocation problems. In *ACM Conference on Electronic Commerce*, pages 29–38, 2011. 2

[11] J. Feldman, M. Henzinger, N. Korula, V. S. Mirrokni, and C. Stein. Online stochastic packing applied to display ad allocation. In *Proceedings of the 18th annual European conference on Algorithms: Part I*, ESA'10, pages 182–194, Berlin, Heidelberg, 2010. Springer-Verlag. 2, 4

[12] J. Feldman, N. Korula, V. Mirrokni, S. Muthukrishnan, and M. Pal. Online ad assignment with free disposal. In *WINE*, 2009. 2, 4, 6

[13] J. Feldman, A. Mehta, V. Mirrokni, and S. Muthukrishnan. Online stochastic matching: Beating 1 - 1/e. In *FOCS*, 2009. 2

[14] B. Kalyanasundaram and K. Pruhs. On-line network optimization problems. In *Developments from a June 1996 seminar on Online algorithms: the state of the art*, pages 268–280, London, UK, 1998. Springer-Verlag. 1, 3, 6

[15] C. Karande, A. Mehta, and P. Tripathi. Online bipartite matching with unknown distributions. In *STOC*, 2011. 4

[16] R. Karp, U. Vazirani, and V. Vazirani. An optimal algorithm for online bipartite matching. In *Proc. STOC*, 1990. 1, 2

[17] M. Mahdian, H. Nazerzadeh, and A. Saberi. Allocating online advertisement space with unreliable estimates. In *ACM Conference on Electronic Commerce*, pages 288–294, 2007. 4, 6

[18] M. Mahdian and Q. Yan. Online bipartite matching with random arrivals: A strongly factor revealing lp approach. In *STOC*, 2011. 3, 4

[19] A. Mehta, A. Saberi, U. Vazirani, and V. Vazirani. Adwords and generalized online matching. *J. ACM*, 54(5):22, 2007. 1, 2, 3, 4, 6

[20] H. Menshadi, S. Oveis Gharan, and A. Saberi. Offline optimization for online stochastic matching. In *SODA*, 2011. 2, 3, 4

[21] R. Motwani, R. Panigrahy, and Y. Xu. Fractional matching via balls-and-bins. In *APPROX-RANDOM*, pages 487–498, 2006. 3

[22] A. Panconesi and A. Srinivasan. Randomized distributed edge coloring via an extension of the chernoff-hoeffding bounds. *Siam Journal on Computing*, 26:350–368, 1997. 7, 15

[23] B. Tan and R. Srikant. Online advertisement, optimization and stochastic networks. *CoRR*, abs/1009.0870, 2010. 2, 4

[24] E. Vee, S. Vassilvitskii, and J. Shanmugasundaram. Optimal online assignment with forecasts. In *ACM EC*, 2010. 2, 4

[25] H. Wang, H. X. 0002, L. Qiu, Y. R. Yang, Y. Zhang, and A. G. Greenberg. Cope: traffic engineering in dynamic networks. In *SIGCOMM*, pages 99–110, 2006. 2, 4