Improved Bounds for Randomized Preemptive Online Matching

Leah Epstein* Asaf Levin† Danny Segev‡ Oren Weimann§

Abstract

Preemptive online algorithms for the *maximum matching* problem maintain a valid matching M while edges of the underlying graph are presented one after the other. When presented with an edge e, the algorithm should decide whether to augment the matching M by adding e (in which case e may be removed later on) or to keep M in its current form without adding e (in which case e is lost for good). The objective is to eventually hold a matching M with maximum weight.

The main contribution of this paper is to establish new lower and upper bounds on the competitive ratio achievable by randomized preemptive online algorithms:

- We provide a lower bound of $1+\ln 2 \approx 1.693$ on the competitive ratio of any randomized algorithm for the maximum cardinality matching problem. This result improves on the currently best known bound of $e/(e-1) \approx 1.581$, implied by the work of Karp, Vazirani, and Vazirani [STOC '90] on a closely-related model with vertex arrivals.
- We devise a randomized algorithm that achieves an expected competitive ratio of 5.356 for maximum weight matching. This finding demonstrates the power of randomization in this context, showing how to beat the tight bound of $3+2\sqrt{2} \approx 5.828$ for deterministic algorithms, obtained by combining the 5.828 upper bound of McGregor [APPROX '05] and the 5.828 lower bound of Varadaraja [ICALP '11].

1 Introduction

In the *maximum matching* problem, we are given an undirected graph $G = (V, E)$ whose edges are associated with non-negative weights. A set of edges $M \subseteq E$ is called a *matching* when no two of them share a common vertex. The objective is to compute a matching of maximum total weight, where the total weight of the output matching is also called the profit of the algorithm. Due to its wide real-life applicability, as well as to its appealing theoretical nature, this computational problem has received a great deal of attention from various communities such as computer science, mathematics, operations research, and economics (see Schrijver's book [30] and references therein for a comprehensive overview of classic work).

As can only be expected, the algorithmic research revolving around maximum matching has expanded from studying the traditional (offline) setting to additional models. In particular, the

*Department of Mathematics, University of Haifa, 31905 Haifa, Israel. Email: lea@math.haifa.ac.il.

†Faculty of Industrial Engineering and Management, The Technion, 32000 Haifa, Israel. Email: levinas@ie.technion.ac.il.

‡Department of Statistics, University of Haifa, 31905 Haifa, Israel. Email: segevd@stat.haifa.ac.il.

§Department of Computer Science, University of Haifa, 31905 Haifa, Israel. Email: oren@cs.haifa.ac.il.

online setting has extensively been studied over the last few decades [4, 13, 14, 23, 24, 25, 26]. In our model, the edges (along with their weights) are presented one by one to the algorithm, which is required to keep a valid matching M at all times. In other words, once an edge e is presented, the algorithm must decide whether to add it to M or not. However, e may be added only if the resulting set of edges $M \cup \{e\}$ remains a valid matching. If the algorithm decides to add the edge e to M, we say that e is *accepted*, and if the algorithm decides not to add e, we say that e is *rejected*. With this regard, to formally specify the online setting being considered, the remaining question is whether the acceptance of an edge is permanent or not.

In the non-preemptive model, the decision of whether or not to add any given edge to M is irrevocable, i.e., once an edge is added to the set M of previously accepted edges it can never be removed. The final matching thus consists of all edges that were ever accepted. Alas, in this model, simple examples demonstrate that the competitive ratio of any (deterministic or randomized) algorithm exceeds any function of the number of vertices, meaning that no competitive algorithm exists (see, for example, [13]). That being said, in the unweighted case (where all edge weights are equal, which is also called the *maximum cardinality matching* problem), a greedy approach that accepts an edge whenever possible has a competitive ratio of 2. For deterministic algorithms, this ratio is actually best possible, as shown by Karp, Vazirani, and Vazirani [24].

In the preemptive model, the algorithm is given more freedom by allowing it to remove previously accepted edges from the current matching at any point in time; this event is called *preemption*. Nevertheless, an edge that was either rejected or preempted cannot be re-inserted to the matching later on. As opposed to the non-preemptive model, with this extra freedom competitive algorithms do exist. Specifically, a deterministic algorithm that was proposed by Feigenbaum et al. [14] attains a competitive ratio of 6. Later on, McGregor [26] improved on this finding, by tweaking it into achieving a ratio of $3 + 2\sqrt{2} \approx 5.828$. On the other hand, Epstein et al. [13] established a lower bound of 4.967 for any deterministic algorithm, which has subsequently been improved by Varadaraja [3] to (a tight bound of) $3 + 2\sqrt{2} \approx 5.828$.

The upper bound of McGregor and the lower bound of Varadaraja establish a tight bound of $3 + 2\sqrt{2} \approx 5.828$ on the competitive ratio of any *deterministic* algorithm in the preemptive model. For *randomized* algorithms, the currently best lower bound of $e/(e-1) \approx 1.581$ can be inferred from the work of Karp et al. [24] on a related model, which is further discussed below. Interestingly, their lower bound proof actually works for the unweighted case.

Semi-streaming matching and other related work. What seems to have ignited renewed interest in the preemptive online model is the investigation of maximum matching in the *semi-streaming* model, which was introduced by Muthukrishnan [27]. On the other hand, online algorithms are a long-standing and classic research field (including preemptive variants such as those of [5, 17, 31, 10, 9]), and the interest in such models is not restricted to its semi-streaming applications. In semi-streaming models, an algorithm performs one or more passes of reading the input as a stream. This means that, in a single pass, it cannot go back and re-examine parts of the input that appear earlier in the stream. The algorithm is allowed, however, to keep a limited amount of information in memory which is based on the input seen thus far. Next,

we discuss a more precise definition which is relevant to matching. In the single-pass version of maximum matching problems in the *semi-streaming* model, which is more relevant for our purposes, the edges (along with their weights) are presented one by one to the algorithm, which is allowed to use only $O(n \cdot \text{polylog}(n))$ space to store information at all times (including a potential output, if it wishes to spend some of its limited memory on that), but is not required to hold a valid matching[1]. For matching problems, the stored information is typically just a set of edges, and the possibility to keep in memory *any* small set of edges (rather than only a matching) is what gives this model its added strength in comparison to the preemptive online model. When multiple passes are allowed, the entire sequence of edges is presented multiple times to the algorithm, allowing it to examine the input again, possibly modifying the set of edges stored in memory, and possibly resulting in an improved output. Below, we discuss the case of a single pass, and refer the interested reader to [22] for recent results and a discussion on multiple-pass matching algorithms (see also McGregor [26] and Ahn and Guha [1, 2]).

Epstein et al. [13] observed that the semi-streaming algorithms of Feigenbaum et al. [14] and McGregor [26] (of approximation ratios 6 and 5.828, respectively) can actually be viewed as preemptive online algorithms in disguise, and this is what suggested a connection between the two variants. However, the semi-streaming model is not as strict as the preemptive online model, and the subsequent algorithms of Zelke [34] and that of Epstein et al. [13] (whose approximation ratios are 5.585 and 4.91, respectively) are *not* deterministic preemptive online algorithms. Specifically, the latter may simultaneously hold $\Omega(\log n)$ matchings in memory, arguing that their union contains a good matching, while the former keeps several additional edges for each edge in the current matching. It is worth pointing out that the lower bound of Varadaraja [3] on the competitive ratio of deterministic one-pass semi-streaming algorithms also shows that those algorithms cannot be converted into deterministic preemptive online algorithms while maintaining the same approximation ratio. The above-mentioned approximation ratios were improved to $4 + \varepsilon$ by Crouch and Stubbs [7], and later on to $3.5 + \varepsilon$ by Grigorescu and Monemizadeh, and Zhou [19], using the same algorithm. All those algorithms use a method called *bucketing* in different ways (which we describe below as our randomized algorithm uses this method), leading to relatively high approximation ratios. Finally, Paz and Schwartzman [29], designed a $(2 + \varepsilon)$-approximation via a clever implementation of the local ratio technique. It is unclear whether it is possible to convert these algorithms for the semi-streaming model into randomized preemptive online algorithms.

Two variants for online matching. We proceed by discussing the differences and similarities between our edge-arrival model and the one-sided vertex-arrival model of Karp et al. [24], which is very relevant for our purposes here. In our model, edges of a general graph are presented one by one. In the one-sided vertex-arrival model, the underlying graph is bipartite, where one part is given in advance, and for the other part, every vertex is revealed together with all its incident edges. Any algorithm for our model can be used in the latter model, simply by processing the edges arriving one by one, rather than simultaneously. Thus, the lower bound of 1.581 due to Karp et al. [24] is valid for the randomized variant of our model. However, in

[1]Here, n stands for the number of vertices in the underlying graph, meaning in particular that, when the latter is sufficiently dense, most edges cannot be kept in memory.

our lower bound proof, while we construct a bipartite graph as well, our construction consists of a layered graph, where layers are revealed one after the other. Thus, our instance is not presented to the online algorithm in the same way as it would have been presented in the one-sided vertex-arrival model. In particular, even though certain sets of edges are created such that they could be presented together, the edges incident to each vertex are not presented at once (or even consecutively).

1.1 Our results

The main contribution of this paper is to establish new lower and upper bounds on the competitive ratio of *randomized* algorithms in the *preemptive online model*. Our findings, along with some technical comments, can be briefly summarized as follows.

A lower bound for unweighted graphs. In Section 2, we provide a novel construction, establishing a lower bound of $1 + \ln 2 \approx 1.693$ on the competitive ratio of any randomized online algorithm for the maximum cardinality matching problem. This result improves on the currently best known bound of $e/(e-1) \approx 1.581$, that can be inferred from the work of Karp et al. [24] on the one-sided vertex-arrival model (as discussed above). It is worth pointing out that the latter has originally been proven to be best possible for their model[2]. Due to the differences in the two models, our lower bound does not contradict the results of Karp et al. [24].

An upper bound for weighted graphs. As previously mentioned, when arriving edges are associate with non-negative weights, there is a tight bound of $3 + 2\sqrt{2} \approx 5.828$ on the competitive ratio of any deterministic algorithm [26, 3]. An interesting open question is whether randomization offers any advantage in this context. In Section 3, we answer this question in the affirmative, by devising a *randomized* preemptive online algorithm that beats the aforementioned bound, and achieves a competitive ratio of $\theta \approx 5.356$, where θ is the unique solution to $2(\ln \theta + 1) = \theta$ over $(2, \infty)$.

1.2 Techniques

For the lower bound we use Yao's principle for profit maximization problems [32]. This allows us to assume that the algorithm is deterministic while inputs are presented based on a probability distribution. The set of input graphs in the support of our distribution are presented in layers. The approach of constructing inputs gradually over some axis is a common approach for this type of lower bound constructions (see, for example, [15, 10]). However, our construction is not directly based on previous work, and its analysis requires using new probabilistic arguments.

For the upper bound, we employ a widely used bucketing approach, where weights are classified into types based on intervals to which they belong. The most basic case of this approach is to round weights up to the closest power of 2. In some cases, improvements can be achieved by allowing this parameter to be a power of some positive real $\gamma > 1$, which is not

[2]Specifically, the authors also proposed the well-known ranking algorithm, whose competitive ratio exactly matches the lower bound of $e/(e-1)$.

necessarily 2. The next step is randomization, with rounding to values of the form $\alpha \cdot \gamma^k$, where α is a random variable taking values in $(1, \gamma]$ and k is an integer (not necessarily a positive one).

The need for such an approach in our setting is based on the following motivation. Generally, one would want to preempt an edge of small weight in favor of an edge with a larger weight, as otherwise the resulting matching will consist of edges of small weight, while an optimal matching will consist of those with large weights. However, allowing to replace an edge by another edge of larger weight could result in too many replacements. For example, with respect to a path $v_0, v_1, \ldots, v_t$, if $w(v_0, v_1) = 1$, and $w(v_i, v_{i+1}) = w(v_{i-1}, v_i) + \delta$ for a sufficiently small $\delta > 0$, presenting the edges ordered by increasing indices of endpoints will result in a matching consisting of the last edge, with weight $1 + (t-1)\delta$, while an optimal matching will contain roughly half of the edges, with weight at least $t/2$.

The approach of Feigenbaum et al. [14] and McGregor [26] (previously used, for example, in [5, 31, 17, 9]) is to allow preemption if the newly revealed edge is significantly heavier than the (at most two) edges preempted in order to add the new edge to the current matching. Alternatively, one can use the rounded values when applying comparison between values, in which case the concept of a larger weight means that the weight is rounded to a larger value (see, e.g., [6, 28]). Then, it is possible to introduce randomization, and randomly choose α for the bucketing by $\alpha \cdot \gamma^k$ values (see, e.g., [18, 21, 16, 8, 10, 11, 12, 20]). We refer to this randomized rounding of the weights as *randomized geometric rounding*.

2 A Lower Bound for Randomized Algorithms

In this section, we establish a lower bound of $1 + \ln 2 \approx 1.693$ on the competitive ratio of any randomized algorithm in the preemptive online setting. For ease of presentation, we use Yao's principle for profit maximization problems [32]. That is, to prove a lower bound of C on the achievable (randomized) competitive ratio, we define a probability distribution on a class of inputs such that any deterministic algorithm (evaluated on this distribution) must have a competitive ratio of at least C in expectation.

2.1 The randomized construction

Prior to delving into technical details, we provide a high-level description of how our construction works, along with some additional intuition. In what follows, the underlying graph in every realization of our randomized construction will be comprised of L layers (or vertical columns), each consisting of $2n$ vertices, as shown in Figure 1. It is instructive to think of L and n as very large integers. With this structure in place, the (random) input sequence starts with a random set of edges connecting vertices in layer 1 to vertices in layer 2. As soon as this sequence terminates, a new one begins, with a random set of edges between layers 2 and 3, then between layers 3 and 4, so on and so forth. The edges between successive layers ℓ and $\ell + 1$ are revealed in rounds: In each round, all the edges connecting a vertex u in layer ℓ to its neighbors in layer $\ell + 1$ are revealed one after the other.

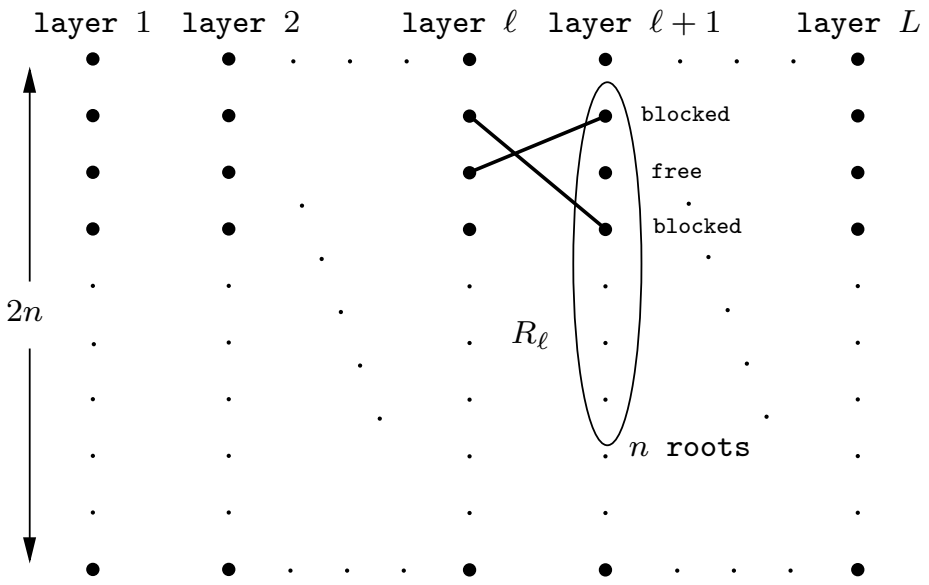

Figure 1: The layered graph used for the lower bound.

Roots. In each layer $1 \leq \ell \leq L-1$, we will designate n out of its $2n$ vertices as being *roots*, collectively forming the set R_ℓ (that is, R_ℓ is the random set of roots in layer ℓ). These roots are a-priori unknown, and will be determined as soon as the random sequence of edges between layers $\ell - 1$ and ℓ terminates. This holds true for any layer other than the first, in which R_1 is defined by deterministically picking n arbitrary vertices.

Root permutation. Once the set of roots R_ℓ is determined, we linearly order R_ℓ by picking a random permutation of these roots, such that each of the $n!$ possible permutations is equally likely. Clearly, for any $r \in R_\ell$, the (random) position of r in this permutation is uniformly distributed over $\{1, \ldots, n\}$.

Revealing the edges. We now explain how the edges between layers ℓ and $\ell+1$ are revealed. Based on the permutation picked earlier, we introduce edges connecting the first root to all $2n$ vertices in layer $\ell + 1$, which are initially colored white. We now choose a random vertex in layer $\ell+1$, which is picked uniformly out of all $2n$ vertices, and color it black. The second root in the linear ordering is then connected to all $2n - 1$ white vertices in layer $\ell + 1$, and out of these $2n - 1$ vertices one is randomly and uniformly chosen to be colored black. This process continues for n iterations, until all roots in R_ℓ have been considered. We still have to define a set of roots $R_{\ell+1}$ for the next layer, and these will be the n white vertices remaining in layer $\ell + 1$. We note that the vertex coloring in layer $\ell + 1$ is not revealed to the online algorithm along the way, i.e., throughout the process of introducing edges between layers ℓ and $\ell+1$. The precise realization of this coloring can be inferred by the algorithm only later on.

2.2 Basic properties

Now let us focus on a fixed deterministic online algorithm ALG and on a fixed realization of our randomized construction. We proceed by establishing two properties, that will be crucial in deriving our main result.

Property 1: No preemption from previous layers. Once ALG picks a subset of edges between layers ℓ and $\ell + 1$, there is no incentive to preempt any of them when we proceed to subsequent layers. This property is achieved due to the fact that edges are revealed layer by layer. To see this, note that if ALG preempts such edge, say (u, v), then either the vertex v is eventually left unmatched, meaning that we have just lost (u, v) and gained nothing in return. Or, the vertex v is matched to a vertex in layer $\ell + 2$, meaning that the number of edges in the current matching remains unchanged, and we have just made things worse further down the road by (tentatively) matching a vertex in layer $\ell + 2$.

Property 2: Every free root is matched at the end of its round, forever. Consider the round in which the edges connecting a root r in layer ℓ to its neighbors in layer $\ell + 1$ are revealed. The root r must be connected to at least two unmatched neighbors in layer $\ell + 1$, since for any round $1 \leq i \leq n$, the current root is connected to $2n - (i - 1)$ vertices, out of which at most $i - 1$ have previously been matched, so the number of unmatched vertices is at least $2n - 2(i - 1) \geq 2$. The first observation we make use of is that, if r had not been matched to a vertex in layer $\ell - 1$, ALG always matches r upon termination of the current round. As previously argued, r can indeed be matched, and ALG has no motivation to leave r unmatched, as the additional edge can be preempted later on.

Interestingly, we proceed by showing that ALG never preempts this edge. Let (r, v) be the edge ALG holds once r's round terminates, where v is some vertex in layer $\ell + 1$. From this point on, there is no motivation to ever preempt (r, v). To verify this claim, note that if ALG preempts (r, v), it has two options:

- Leave v unmatched to other roots in layer ℓ, meaning that we have just lost (r, v) and gained nothing, since the only possible reason to preempt this edge is to match between v and vertices in layer $\ell+2$, but there is no motivation to do so, as argued in our explanation for Property 1.
- Match v to some other root in layer ℓ, which leaves us at exactly the same situation when proceeding to the next layer (i.e., v is still matched).

Remark. We mention in passing that, by Properties 1 and 2, it may seem as if preemptions do not occur at all. However, this is certainly not the case, as preemptions can occur within a round. That is, while the edges connecting a root r in layer ℓ to all its neighbors in layer $\ell + 1$ are revealed, ALG may preempt a previously chosen edge (r, v) in favor of a different one, (r, v').

2.3 Analysis

The preceding discussion will allow us to focus on deterministic online algorithms that pick edges in a very particular way. In fact, we are able to prove an explicit upper bound of $\frac{1}{1+\ln 2} \cdot Ln + O(L)$ on the expected cardinality of the matching picked by ALG. On the other hand, in any realization of the randomized construction, we will show that there exists a feasible matching of cardinality $(L-1)n$. These observations lead to the next theorem.

Theorem 2.1. *The competitive ratio of any randomized preemptive online algorithm for maximum cardinality matching is at least* $1 + \ln 2$.

We begin by utilizing the specific random process according to which edges are revealed in order to compare between what can be achieved with and without knowing the realization of the input sequence in advance. For this purpose, we start off with the easier part, arguing that the random graph created always contains a matching of cardinality $(L-1)n$, as stated in the next claim.

Lemma 2.2. *For any realization of our randomized construction, the maximum cardinality of a matching is at least* $(L-1)n$.

Proof. Fix a realization of our construction. In order to obtain a matching consisting of $(L-1)n$ edges, for every layer $1 \leq \ell \leq L-1$ one can pick a matching of size n between the roots R_ℓ and the vertices in layer $\ell+1$ that ended up being colored black. A matching of this nature necessarily exists since the first root can be matched to the first vertex being colored black, the second root to the second vertex being colored black, and so on. □

The challenging part is that of bounding the expected cardinality of the matching computed by a fixed deterministic online algorithm ALG that satisfies Properties 1 and 2, where the expectation is taken over the random choices in our construction. To this end, upon termination of the phase where edges between layers $\ell-1$ and ℓ are introduced, we say that a root in R_ℓ is *free* when it has not been matched by ALG to some vertex in the preceding layer $\ell-1$, as illustrated in Figure 1. Otherwise, this root is said to be *blocked*. For $1 \leq \ell \leq L-1$, let F_ℓ and B_ℓ be random variables that stand for the number of free and blocked roots in R_ℓ, respectively. The next lemma relates between these random variables and the number of edges picked by ALG.

Lemma 2.3. *ALG computes a (random) matching of cardinality* $\sum_{\ell=1}^{L-1} F_\ell$.

Proof. Fix a realization of R_ℓ, and let us focus on some root $r \in R_\ell$. Consider the random process in which edges between layers ℓ and $\ell+1$ are revealed, formally described in Section 2.1:

- If the root r is blocked, none of the edges introduced between r and its neighbors in level $\ell+1$ is picked by ALG, or otherwise, the edge that connects r to a vertex in level $\ell-1$ has to be preempted, contradicting Property 1.
- In the opposite case, where r is free, ALG will expand the matching it holds by adding a new edge adjacent to this root, and that edge will not be preempted in the future, by Property 2.

To summarize, when F_ℓ out of n roots in R_ℓ are free, ALG will add exactly F_ℓ edges to the matching (out of those connecting layer ℓ to $\ell+1$; each adjacent to a root in F_ℓ), which will not be preempted later on. This applies to any realization of R_ℓ. Therefore, the random cardinality of the matching computed is $\sum_{\ell=1}^{L-1} F_\ell$. □

Recursively computing $\mathbb{E}[F_\ell]$. In what follows, we derive a recursive inequality that relates between the expected number of free roots in successive layers. For this purpose, we look into the question of how free roots (or equivalently, blocked roots) are created. In particular, we argue that the random permutation by which roots are processed guarantees that each free root in the current layer will create a (distinct) blocked root in the next layer with probability $\ln 2 + \Theta(\frac{1}{n})$.

Lemma 2.4. *There exists a layer-independent constant $c > 0$, such that for every $1 \leq \ell \leq L-1$,*

$$|\mathbb{E}\left[F_{\ell+1}\right] - n + \ln 2 \cdot \mathbb{E}\left[F_\ell\right]| \leq c \ .$$

Proof. Since $F_{\ell+1}$ and $B_{\ell+1}$ form a partition of the roots $R_{\ell+1}$, we have in particular $\mathbb{E}[F_{\ell+1}] = n - \mathbb{E}[B_{\ell+1}]$. Therefore, it is sufficient to prove that $|\mathbb{E}[B_{\ell+1}] - \ln 2 \cdot \mathbb{E}\left[F_\ell\right]| \leq c$, for an absolute constant $c > 0$, independent of ℓ. To this end, since $\mathbb{E}[B_{\ell+1}] = \mathbb{E}[\mathbb{E}[B_{\ell+1}|F_\ell]]$, we will show that $\mathbb{E}[B_{\ell+1}|F_\ell = k] = \ln 2 \cdot k + \Theta(\frac{k}{n})$, i.e., given that there are k free roots in layer ℓ, the expected number of blocked roots in layer $\ell+1$ is $\ln 2 \cdot k$, up to some additive constant.

To this end, recall that, by Property 2, each free root in F_ℓ is necessarily matched by ALG to a vertex in layer $\ell+1$, and these edges will not be preempted later on. Based on this property, for $1 \leq p \leq n$, let $I_{\ell,p}$ be an indicator variable for the event where position p in the random permutation picked for R_ℓ is occupied by a free root and, in addition, the vertex in level $\ell+1$ to which this free root is matched survives the random recoloring step in all subsequent iterations as a white vertex. As these are precisely the events by which blocked roots in layer $\ell+1$ are created, it follows that $B_{\ell+1} = \sum_{p=1}^{n} I_{\ell,p}$, and therefore

$$\mathbb{E}\left[B_{\ell+1}|F_\ell = k\right] = \sum_{p=1}^{n} \mathbb{E}\left[I_{\ell,p}|F_\ell = k\right] = \sum_{p=1}^{n} \Pr\left[I_{\ell,p} = 1|F_\ell = k\right] \ .$$

In order to compute $\Pr\left[I_{\ell,p} = 1|F_\ell = k\right]$, note that, conditional on the event $[F_\ell = k]$, position p in the random permutation of R_ℓ is occupied by a free root with probability $\frac{k}{n}$. The important observation is that, conditional on the event where position p is occupied by a free root, the event where the vertex in level $\ell+1$ to which this free root is matched survives as a white vertex is independent of $[F_\ell = k]$. Specifically, since there are $n-p$ remaining rounds (excluding the current one), the latter survival probability is $\frac{n}{2n-p+1}$, as out of the $2n-p+1$ currently white vertices, every subset of n vertices has equal probability to be the set of vertices that remain white. Consequently, we have $\Pr\left[I_{\ell,p} = 1|F_\ell = k\right] = \frac{k}{n} \cdot \frac{n}{2n-p+1} = \frac{k}{2n-p+1}$, implying that

$$\mathbb{E}\left[B_{\ell+1}|F_\ell = k\right] = k \cdot \sum_{p=1}^{n} \frac{1}{2n-p+1} = k \cdot (H_{2n} - H_n) = \ln 2 \cdot k + \Theta\left(\frac{k}{n}\right) \ ,$$

where for a positive integer t, $H_t = 1 + \frac{1}{2} + \cdots + \frac{1}{t}$ denotes the t-th harmonic number. The last equation holds since $|H_t - \ln t| = \gamma + O(\frac{1}{t})$, where γ is the Euler-Mascheroni constant [33]. $\square$

Concluding the proof of Theorem 2.1. Recall that by Lemma 2.4, we have in particular $\mathbb{E}[F_{\ell+1}] + \ln 2 \cdot \mathbb{E}[F_\ell] \leq n + c$ for $1 \leq \ell \leq L-1$. Summing these inequalities over all values of ℓ gives

$$\begin{aligned}(L-1)\cdot(n+c) &\geq \sum_{\ell=1}^{L-1} \mathbb{E}\left[F_{\ell+1}\right] + \ln 2 \cdot \sum_{\ell=1}^{L-1} \mathbb{E}\left[F_\ell\right] \\ &= (1+\ln 2)\cdot \sum_{\ell=1}^{L-1} \mathbb{E}\left[F_\ell\right] + \mathbb{E}\left[F_L\right] - \mathbb{E}\left[F_1\right] \\ &\geq (1+\ln 2)\cdot \sum_{\ell=1}^{L-1} \mathbb{E}\left[F_\ell\right] - n \ ,\end{aligned}$$

where the last inequality holds since $F_L \geq 0$ and $F_1 = n$. Consequently, by Lemma 2.3 the expected cardinality of the matching computed by ALG is

$$\sum_{\ell=1}^{L-1} \mathbb{E}\left[F_\ell\right] \leq \frac{1}{1+\ln 2} \cdot (Ln + (L-1)c) \ .$$

On the other hand, the maximum cardinality of a matching in every realization of the resulting graph is $(L-1)n$, as shown in Lemma 2.2, implying that the asymptotic competitive ratio of ALG (when L and n tend to infinity) is at least $1 + \ln 2$.

3 A Randomized Algorithm

In this section, we show that by employing randomization, the lower bound of $3 + 2\sqrt{2} \approx 5.828$ on the performance of any deterministic algorithm can be beaten. In particular, by making use of randomized geometric rounding (see Section 1.2), our algorithm achieves an expected competitive ratio of roughly 5.356.

3.1 The algorithm

Parameters. In what follows, we utilize two real-valued parameters: A *base* $\theta > 1$, and a *shifting value* $\phi > 0$. The base θ will be optimized later on, so that the resulting upper bound on the competitive ratio of our algorithm is made as small as possible. In contrast, the shifting value ϕ will not be fixed; instead, it will be a random variable whose value is chosen to be θ^τ, where τ is uniformly distributed over the interval $(0, 1]$.

Weight classes and rounded weights. We define weight classes of edges in the following way. Let $w(e)$ denote the (non-negative) weight of an edge e. For every $i \in \mathbb{Z}$, we let the weight class W_i be the collection of edges whose weight is in the interval $[\phi\theta^i, \phi\theta^{i+1})$. Observe that the set of weight classes is a partition of the edges of strictly positive weight. Once an edge e is presented, if its weight is positive, then we round down its original weight $w(e)$ to the

lower endpoint of the weight class it belongs to, thereby obtaining its *rounded* weight $\tilde{w}(e)$. In other words, letting i be the unique integer for which $w(e) \in [\phi\theta^i, \phi\theta^{i+1})$, we set $\tilde{w}(e) = \phi\theta^i$. In the remainder of this section, the latter notation will be extended to matchings, so that $w(M)$ and $\tilde{w}(M)$ will stand for the (total) original weight and rounded weight, respectively, of a matching M. In addition, for a maximum weight matching M^*, we denote $\mathrm{OPT} = w(M^*)$ and $\widetilde{\mathrm{OPT}} = \tilde{w}(M^*)$. Moreover, let $\widetilde{\mathrm{OPT}}_\tau$ denote the profit of a maximum weight matching for the weight function $\tilde{w}$ resulting from a particular realization of τ.

Maintaining a matching online. The algorithm keeps a tentative matching M, which is initialized prior to reading the input sequence as $M = \emptyset$. Upon the arrival of a newly-presented edge $e = (u, v)$, we proceed as follows. If $w(e) = 0$, we reject e. Otherwise, let $X(M, e)$ denote the set of edges in the matching M that have a common endpoint with e, that is, edges that have u or v as an endpoint. Note that $X(M, e)$ is a random set, consisting of at most two edges. Now, if every edge $e' \in X(M, e)$ satisfies $\tilde{w}(e') < \tilde{w}(e)$ then e is inserted into M while the edges in $X(M, e)$ are preempted, i.e., we set $M \leftarrow (M \setminus X(M, e)) \cup \{e\}$. Otherwise, M remains unchanged.

3.2 Analysis

Without loss of generality, we assume that every edge e satisfies $w(e) > 0$. We can indeed make this assumption, as zero weight edges can be removed from M^* and the algorithm discards such edges.

We begin by accounting for the extent to which the weight of each edge is rounded. Specifically, the next lemma shows how to evaluate the ratio between the expected rounded weight of any edge[3] to its original weight in terms of the base θ. Subsequently, this allows us to bound the ratio between OPT and $\widetilde{\mathrm{OPT}}$.

Lemma 3.1. *For every edge e, we have $\mathbb{E}_\tau[\tilde{w}(e)/w(e)] = \frac{\theta-1}{\theta \ln \theta}$.*

Proof. We denote by $\tilde{w}^\tau(e)$ the value $\tilde{w}(e)$ for a given realization of τ. Let p be the unique integer and let $0 < \alpha \leq 1$ be the unique real for which $w(e) = \theta^{p+\alpha}$. By our definition of the weight classes $\{W_i\}_{i \in \mathbb{Z}}$, it follows that the rounded weight $\tilde{w}^\tau(e)$ is determined by:

$$\tilde{w}^\tau(e) = \begin{cases} \theta^{p+\tau} & \text{if } \tau \leq \alpha \\ \theta^{p+\tau-1} & \text{if } \tau > \alpha \end{cases}$$

Therefore, the ratio between the expected rounded weight of e to its original weight is

$$\mathbb{E}_\tau\left[\frac{\tilde{w}(e)}{w(e)}\right] = \int_0^\alpha \frac{\theta^{p+\tau}}{\theta^{p+\alpha}} d\tau + \int_\alpha^1 \frac{\theta^{p+\tau-1}}{\theta^{p+\alpha}} d\tau = \frac{1}{\ln \theta} \cdot \left(\frac{1}{\theta^\alpha}\left(\theta^\alpha - 1\right) + \frac{1}{\theta^{\alpha+1}}\left(\theta - \theta^\alpha\right)\right) = \frac{\theta - 1}{\theta \ln \theta} \,.$$

□

Lemma 3.2. *The expected rounded weight of the optimal matching M^* satisfies $\mathbb{E}_\tau[\widetilde{\mathrm{OPT}}] = \frac{\theta-1}{\theta \ln \theta}\mathrm{OPT}$. Also, for any realization of the random variable τ, we have $\mathrm{OPT} < \theta \cdot \widetilde{\mathrm{OPT}}_\tau$.*

[3]Note that the expectation $\mathbb{E}_\tau[\cdot]$ is taken over the random choice of the variable $\tau \sim U(0, 1]$.

Proof. It is easy to verify that, due to linearity of expectation, the first claim follows from separately applying Lemma 3.1 on every edge of the optimal matching M^*, and summing over all edges. On the other hand, the second claim holds since, regardless of the realization of τ, for any edge e (in particular, those in M^*) we have $\tilde{w}(e)/w(e) > 1/\theta$, as the lower and upper endpoints of each weight class differ by a multiplicative factor of at most θ. □

Up until now we have merely discussed how the expected weight of the optimal matching M^* depends on the geometric rounding procedure. We now move on to describe the technical crux in our analysis, where the online matching computed by the algorithm is compared against the rounded weight of M^*.

Lemma 3.3. *For any realization of τ, the total weight obtained by the algorithm is at least $\frac{\theta-2}{2\theta-2}\widetilde{\mathrm{OPT}}$.*

Proof. We consider how the algorithm operates for an arbitrary realization of the variable τ. For this purpose, consider an optimal solution $\tilde{M}$ for the *rounded* instance, i.e., a maximum weight matching with respect to the rounded weights $\tilde{w}(\cdot)$. Clearly, $\tilde{w}(\tilde{M}) \geq \tilde{w}(M^*) = \widetilde{\mathrm{OPT}}$, as M^* is an optimal matching for the original weights, but not necessarily for the rounded weights.

We map every edge of $\tilde{M}$ to an edge in the matching M that is being held by the algorithm upon termination, possibly using a single edge in M as the image of several edges in $\tilde{M}$. The way this mapping will be defined below enables us to argue that the ratio between the total rounded weight of the edges mapped to any edge $e \in M$ and between $\tilde{w}(e)$ is at most $\frac{2\theta-2}{\theta-2}$. This claim immediately leads to a ratio of $\frac{2\theta-2}{\theta-2}$ between $\widetilde{\mathrm{OPT}}$ and the total profit of the algorithm according to the rounded weights $\tilde{w}(\cdot)$. Since $w(e) \geq \tilde{w}(e)$ for every edge e, the actual profit of the algorithm can only be higher.

For every edge $e \in M$, we create a *preemption tree*. The root of this tree is e, and the children of an edge e' are all edges that were preempted from the matching kept by the algorithm when e' was inserted. Note that the number of children is either 2, 1, or 0, since any matching cannot contain more than a single edge for every endpoint of e' (prior to the arrival of e'). The set of edges that do not belong to any preemption tree are edges that the algorithm never accepted, while the union of all edge sets over all trees is exactly the set of edges that belonged to the matching at some point in time.

By definition of our algorithm, for every edge $\tilde{e}$ that does not belong to a tree, there exists an edge e' that belongs to a tree, sharing an endpoint with $\tilde{e}$, such that $\tilde{w}(e') \geq \tilde{w}(\tilde{e})$. If, for an edge $\tilde{e} \in \tilde{M}$ such that $\tilde{e}$ does not belong to a tree, there exists a unique such edge e', then $\tilde{e}$ is mapped to the root of the tree where e' appears. Otherwise, one such edge e' is chosen arbitrarily, and $\tilde{e}$ is mapped to the root of the tree of e' in this case as well. For an edge $\tilde{e} \in \tilde{M}$ that belongs to some tree, $\tilde{e}$ is mapped to the root of the tree in which it appears.

Next, consider a specific tree with root $e \in M$. For an edge $\hat{e}$ of distance $d[\hat{e}]$ from the root, measured in number of edges of the preemption tree (and not edges of the input graph), let $\tilde{w}_d(\hat{e}) = \tilde{w}(e)/\theta^{d[\hat{e}]}$. For such an edge $\hat{e}$ of distance $d[\hat{e}]$ from the root, we say that the *level* of $\hat{e}$ is $d[\hat{e}]$. In the next claim, we show that the rounded weight of edges in the tree decreases exponentially in the level.

Claim 3.4. *For every edge $\hat{e}$ in the tree, $\tilde{w}(\hat{e}) \leq \tilde{w}_d(\hat{e})$.*

Proof. Consider the path $e_0, e_1, e_2, \ldots, e_{d[\hat{e}]}$, where $e_0 = \hat{e}$, $e_{d[\hat{e}]} = e$, and for every $1 \leq i \leq d[\hat{e}]$, e_i is the edge whose arrival caused e_{i-1} to be preempted. By definition of the algorithm, $\tilde{w}(e_i) > \tilde{w}(e_{i-1})$. Since our geometric rounding method guarantees that, when rounded weights are not identical, they differ by a multiplicative factor of at least θ, it follows that $\tilde{w}(e_i) \geq \theta \tilde{w}(e_{i-1})$. Therefore, $\tilde{w}(e) = \tilde{w}(e_{d[\hat{e}]}) \geq \theta^{d[\hat{e}]} \tilde{w}(e_0) = \theta^{d[\hat{e}]} \tilde{w}(\hat{e})$, or alternatively $\tilde{w}(\hat{e}) \leq \tilde{w}_d(\hat{e})$. □

For a vertex v in the graph, and for every possible distance $d \geq 0$, we say that v is *new* for d if: (1) there is an edge of level d in some preemption tree for which v is an endpoint; and (2) v is not an endpoint of any edge of smaller level. Also, the two endpoints of an edge $e \in M$ are new for $d = 0$ and are not new for any other value of d.

Claim 3.5. *Fix one preemption tree. For every $d > 0$, there are at most 2^d new vertices for d. Moreover, every edge in a tree in level $d > 0$ has at most one new vertex for d.*

Proof. Since every edge has at most two children, the number of edges of level d in the preemption tree is at most 2^d. Every such edge has a common endpoint with an edge of level $d-1$, thus there is at most one new vertex for d per edge, which results in a total of at most 2^d new vertices for d. □

Claim 3.6. *For every edge $e \in M$, the total rounded weight of the edges mapped to e is at most $\frac{2\theta-2}{\theta-2}\tilde{w}(e)$.*

Proof. For an edge $\tilde{e} \in \tilde{M}$, let $d_{\min}[\tilde{e}]$ be the minimum level such that an edge of this level has a common vertex with $\tilde{e}$. By definition, this common vertex must be new for level $d_{\min}[\tilde{e}]$. Let $\bar{e}$ be the edge of level $d_{\min}[\tilde{e}]$ with the common vertex. Since there exists an edge in the tree having a common endpoint with $\tilde{e}$ of rounded weight at least $\tilde{w}(\tilde{e})$, we have $\tilde{w}(\bar{e}) \geq \tilde{w}(\tilde{e})$. In addition, since $\tilde{M}$ is a matching, for every edge e'' in the tree and every new vertex v of the level of e'' that is incident to e'', at most one edge of $\tilde{M}$ has v as an endpoint. By Claim 3.5, every edge of the tree other than e has at most one new vertex, so the total rounded weight is at most the total rounded weight of all edges in the tree plus $\tilde{w}(e)$. Letting $\mathcal{T}(e)$ denote the set of edges in the tree rooted at e, by Claim 3.4 it follows that the total rounded weight of edges mapped to e is at most

$$\left(\sum_{\bar{e} \in \mathcal{T}(e)} \tilde{w}(\bar{e})\right) + \tilde{w}(e) \leq \left(\sum_{\bar{e} \in \mathcal{T}(e)} \tilde{w}_d(\bar{e})\right) + \tilde{w}(e) \leq \left(\sum_{d=0}^{\infty} 2^d \cdot \frac{\tilde{w}(e)}{\theta^d}\right) + \tilde{w}(e) = \left(\frac{1}{1-2/\theta} + 1\right) \tilde{w}(e) \,.$$

□

This completes the proof of Lemma 3.3. □

Theorem 3.7. *The algorithm achieves an expected competitive ratio of at most $\frac{2\theta \ln \theta}{\theta-2}$. This ratio is minimized for $\theta^* \approx 5.356$, where θ^* is the unique solution to $2(\ln \theta + 1) = \theta$ over $(2, \infty)$, in which case the value of the upper bound on the expected competitive ratio is θ^*.*

Proof. By Lemmas 3.2 and 3.3, we have

$$\mathbb{E}_\tau \left[w(M)\right] \geq \frac{\theta-2}{2\theta-2} \mathbb{E}_\tau \left[\widetilde{\mathrm{OPT}}\right] \geq \frac{\theta-2}{2\theta-2} \cdot \frac{\theta-1}{\theta \ln \theta} \mathrm{OPT} = \frac{\theta-2}{2\theta \ln \theta} \mathrm{OPT} \,.$$

□

References

[1] K. J. Ahn and S. Guha. Laminar families and metric embeddings: Non-bipartite maximum matching problem in the semi-streaming model. Available online at: `http://arxiv.org/abs/1104.4058`.

[2] K. J. Ahn and S. Guha. Linear programming in the semi-streaming model with application to the maximum matching problem. *Information and Computation*, 222:59–79, 2013.

[3] A. Badanidiyuru Varadaraja. Buyback problem – approximate matroid intersection with cancellation costs. In *Proceedings of the 38th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 379–390, 2011.

[4] N. Bansal, N. Buchbinder, A. Gupta, and J. Naor. A randomized $O(\log^2 k)$-competitive algorithm for metric bipartite matching. *Algorithmica*, 68(2):390–403, 2014.

[5] S. K. Baruah, G. Koren, D. Mao, B. Mishra, A. Raghunathan, L. E. Rosier, D. E. Shasha, and F. Wang. On the competitiveness of on-line real-time task scheduling. *Real-Time Systems*, 4(2):125–144, 1992.

[6] M. Chrobak, C. Kenyon, J. Noga, and N. E. Young. Incremental medians via online bidding. *Algorithmica*, 50(4):455–478, 2008.

[7] M. Crouch and D. S. Stubbs. Improved streaming algorithms for weighted matching, via unweighted matching. In *Proceedings of the 17th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*, pages 96–104, 2014.

[8] L. Epstein, M. M. Halldórsson, A. Levin, and H. Shachnai. Weighted sum coloring in batch scheduling of conflicting jobs. *Algorithmica*, 55(4):643–665, 2009.

[9] L. Epstein, L. Jeż, J. Sgall, and R. van Stee. Online scheduling of jobs with fixed start times on related machines. *Algorithmica*, 74(1):156–176, 2016.

[10] L. Epstein and A. Levin. Improved randomized results for the interval selection problem. *Theoretical Computer Science*, 411(34-36):3129–3135, 2010.

[11] L. Epstein and A. Levin. Randomized algorithms for online bounded bidding. *Information Processing Letters*, 110(12-13):503–506, 2010.

[12] L. Epstein and A. Levin. On the max coloring problem. *Theoretical Computer Science*, 462:23–38, 2012.

[13] L. Epstein, A. Levin, J. Mestre, and D. Segev. Improved approximation guarantees for weighted matching in the semi-streaming model. *SIAM Journal on Discrete Mathematics*, 25(3):1251–1265, 2011.

[14] J. Feigenbaum, S. Kannan, A. McGregor, S. Suri, and J. Zhang. On graph problems in a semi-streaming model. *Theoretical Computer Science*, 348(2-3):207–216, 2005.

[15] S. P. Y. Fung, C. K. Poon, and F. Zheng. Online interval scheduling: randomized and multiprocessor cases. *Journal of Combinatorial Optimization*, 16(3):248–262, 2008.

[16] R. Gandhi, M. M. Halldórsson, G. Kortsarz, and H. Shachnai. Improved bounds for scheduling conflicting jobs with minsum criteria. *ACM Transactions on Algorithms*, 4(1), 2008. Article number 11.

[17] J. A. Garay, I. S. Gopal, S. Kutten, Y. Mansour, and M. Yung. Efficient on-line call control algorithms. *Journal of Algorithms*, 23(1):180–194, 1997.

[18] M. X. Goemans and J. M. Kleinberg. An improved approximation ratio for the minimum latency problem. *Mathematical Programming*, 82:111–124, 1998.

[19] E. Grigorescu, M. Monemizadeh, and S. Zhou. Streaming weighted matchings: Optimal meets greedy, 2016. Available online at: `https://arxiv.org/abs/1608.01487`.

[20] M. T. Hajiaghayi, R. Khandekar, G. Kortsarz, and V. Liaghat. On a local protocol for concurrent file transfers. *Theoretical Computer Science*, 55(3):613–636, 2014.

[21] M. M. Halldórsson, G. Kortsarz, and H. Shachnai. Sum coloring interval and k-claw free graphs with application to scheduling dependent jobs. *Algorithmica*, 37(3):187–209, 2003.

[22] S. Kale, S. Tirodkar, and S. Vishwanathan. Maximum matching in two, three, and a few more passes over graph streams, 2017. Available online at: `http://arxiv.org/abs/1702.02559`.

[23] B. Kalyanasundaram and K. Pruhs. Online weighted matching. *Journal of Algorithms*, 14(3):478–488, 1993.

[24] R. Karp, U. Vazirani, and 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.

[25] S. Khuller, S. Mitchell, and V. Vazirani. On-line algorithms for weighted bipartite matching and stable marriages. *Theoretical Computer Science*, 127(2):255–267, 1994.

[26] A. McGregor. Finding graph matchings in data streams. In *Proceedings of the 8th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*, pages 170–181, 2005.

[27] S. Muthukrishnan. *Data Streams: Algorithms and Applications*. Foundations and Trends in Theoretical Computer Science. Now Publishers Inc, 2005.

[28] J. J. Paulus, D. Ye, and G. Zhang. Optimal online-list batch scheduling. *Information Processing Letters*, 109(19):1125–1128, 2009.

[29] A. Paz and G. Schwartzman. A $(2+\epsilon)$-approximation for maximum weight matching in the semi-streaming model. In *Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2153–2161, 2017.

[30] A. Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*, volume 24 of *Algorithms and Combinatorics*. Springer, 2003.

[31] G. J. Woeginger. On-line scheduling of jobs with fixed start and end times. *Theoretical Computer Science*, 130(1):5–16, 1994.

[32] A. C.-C. Yao. Probabilistic computations: Toward a unified measure of complexity (extended abstract). In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 222–227, 1977.

[33] R. M. Young. Euler's constant. *The Mathematical gazette*, 75:187–190, 1991.

[34] M. Zelke. Weighted matching in the semi-streaming model. *Algorithmica*, 62(1-2):1–20, 2012.