On Conceptually Simple Algorithms for Variants of Online Bipartite Matching

Allan Borodin[1]*, Denis Pankratov[1]*, and Amirali Salehi-Abari[2]**

[1] University of Toronto, Toronto ON, Canada
bor@cs.toronto.edu, denisp@cs.toronto.edu
[2] Faculty of Business and IT, UOIT, Oshawa ON, Canada
abari@uoit.ca

Abstract. We present a series of results regarding conceptually simple algorithms for bipartite matching in various online and related models. We first consider a deterministic adversarial model. The best approximation ratio possible for a single-pass deterministic online algorithm is 1/2, which is achieved by any greedy algorithm. Dürr et al. [15] recently presented a 2-pass algorithm called Category-Advice that achieves approximation ratio 3/5. We extend their algorithm to multiple passes. We prove the exact approximation ratio for the k-pass Category-Advice algorithm for all $k \geq 1$, and show that the approximation ratio converges to the inverse of the golden ratio $2/(1+\sqrt{5}) \approx 0.618$ as k goes to infinity. The convergence is extremely fast — the 5-pass Category-Advice algorithm is already within 0.01% of the inverse of the golden ratio. We then consider two natural greedy algorithms —MinDegree and MinRanking. We analyze MinDegree in the online IID model and show that its approximation ratio is exactly $1-1/e$. We analyze MinRanking in the priority model and show that this natural algorithm cannot obtain the approximation of the Ranking algorithm in the ROM model.

Keywords: online bipartite matching · adversarial model · online IID model · priority model · Category-Advice algorithm · MinGreedy algorithm

1 Introduction

Maximum bipartite matching (MBM) is a classical graph problem. Let $G = (U, V, E)$ be a bipartite graph, where U and V are the vertices on the two sides, and $E \subseteq U \times V$ is a set of m edges. The celebrated $O(m\sqrt{n})$ Hopcroft-Karp algorithm [21] was discovered in 1973 where n is the number of vertices. The first improvement in the regime of relatively sparse graphs came forty years later when Madry [26] developed a $\widetilde{O}(m^{10/7})$ algorithm based on electrical flows. For dense graphs, i.e., when $m \approx n^2$, Mucha and Sankowski [31] describe an

* Research is supported by NSERC.
** Research was done while the author was a postdoctoral fellow at the University of Toronto. Research was also supported by NSERC.

algorithm running in time $O(n^\omega)$, where $\omega \leq 2.373$ is the matrix multiplication constant. We refer the interested reader to [27] and [14] and references therein for more information on MBM in the offline setting. While current algorithms for solving MBM optimally in the offline setting are reasonably efficient, they still fall short of linear time algorithms. For large graphs, linear or near linear time algorithms might be required. In that regard, a $(1-\epsilon)$-approximation can be computed in $O(m/\epsilon)$ time by a version of the Hopcroft-Karp algorithm in the offline setting [14]. Arguably, such algorithms are not conceptually simple and require a reasonable understanding of the problem.

Our focus in this paper[3] is on "conceptually simple algorithms" with regard to variants of the online MBM problem. We will not define "conceptual simplicity" but claim that certain types of algorithms (e.g., greedy and local search) usually fall within this informal "you know it when you see it" concept. The online model is sometimes necessitated by applications, and can be studied with respect to a completely adversarial model, the random order model (ROM), or a distributional input model (e.g., known and unknown IID input models). In these models, the algorithm has no control over the ordering of the inputs and must make irrevocable decisions for each input item as it arrives. As such, online algorithms are a prime example of a conceptually simple algorithmic paradigm that can be extended in various ways leading to simple offline algorithms. These online extensions can provide improved performance both in terms of worst-case approximation ratios and in terms of performance on real data. See, for example, the experimental analysis of MaxSat provided by Poloczek and Williamson [35]. This still begs the question as to why we should restrict ourselves to conceptually simple algorithms when better offline algorithms are known.

While conceptually simple algorithms may not match the best approximations realized by more complex methods, they are usually very efficient (i.e., linear or near linear time with small constant factors) and often work well on realistic data exceeding worst-case approximation bounds. Conceptually simple algorithms can also be used as a preprocessing step for initializing a local search algorithm as in Chandra and Halldórsson [11]. Moreover, conceptually simple algorithms are easy to implement and modify with relatively little knowledge about the problem domain. Conceptual simplicity is arguably the main reason for the use of simple mechanisms in auctions (see, for example, Lucier and Syrgkanis [25]) and the success of MapReduce [13] in distributed parallel applications.

We will consider two departures from the adversarial and distributional one-pass online models. In the first departure, we consider a multi-pass online algorithm generalizing the two-pass algorithm in Dürr et al. [15]. In this regard we are also motivated by the Poloczek et al. [34] two-pass algorithm for MaxSat. These multi-pass algorithms can be viewed as de-randomizations of known online algorithms and the Dürr et al. algorithm (and our extension) can also be viewed as an $O(n)$ space semi-streaming algorithm. The second departure is that of priority algorithms [8], a model for greedy and more generally myopic algo-

[3] The full version of the paper can be found on arXiv [9].

rithms that extend online algorithms by allowing the algorithm to determine (in some well-defined way) the order of input arrivals. Other conceptually simple generalizations of the traditional online model are clearly possible, such as the ability to modify previous decisions and parallel executions of online algorithms.

Adversarial Online Model. In 1990, Karp, Vazirani, and Vazirani [24] studied MBM in the adversarial online setting. Any greedy algorithm (yielding a maximal matching) achieves a $1/2$ approximation and no deterministic algorithm can do asymptotically better in the adversarial online model. Karp et al. gave a randomized online algorithm called Ranking and showed that it achieves a $1-1/e \approx 0.632$ expected approximation ratio. Moreover, they proved that no randomized algorithm can beat $1-1/e$ in the adversarial online model. After 17 years, a mistake in the analysis of Ranking was found independently by Krohn and Varadarajan and by Goel and Mehta. A correct proof was first given by Goel and Mehta [19], followed by many alternative proofs.

Online Stochastic Models. Feldman et al. [17] introduced the known IID model for MBM. Feldman et al. model statistics about the upcoming queries by the notion of a type graph $G = (U, V, E)$ and a probability distribution $p : U \to [0, 1]$. The online nodes are sampled from p independently one at a time. An algorithm knows G and p beforehand. As before, an algorithm is required to make an irrevocable decision on how to match each arriving online node. In this setting, the adversary can only choose the type graph and the distribution p but doesn't have further control over the online sequence of nodes. Feldman et al. [17] provide an algorithm beating the $1-1/e$ barrier achieving approximation ratio ≈ 0.67. This work was followed by a long line of work including [3,10,20,22,29]. So far, the best approximation ratio for arbitrary arrival rates is ≈ 0.706 due to Jaillet and Lu [22].

Semi-streaming Model. Streaming algorithms are motivated by the necessity to process extremely large data streams. Much of the streaming literature concerns various forms of counting and statistics gathering. Semi-streaming algorithms are streaming algorithms designed for (say) graph search and optimization problems where the output itself requires $\Omega(n)$ space and hence a realistic goal is to maintain $\tilde{O}(n)$ space rather than space $O(m)$. Eggert et al. [16] provide a FPTAS multi-pass semi-streaming algorithm for MBM using space $\tilde{O}(n)$ in the edge input model. In the vertex input semi-streaming model, Goel et al. [18] give a *deterministic* $1 - 1/e$ approximation and Kapralov [23] proves that no semi-streaming algorithm can improve upon this ratio. (See also the recent survey by McGregor [30].) Streaming algorithms do not have to make online decisions but must maintain small space throughout the computation while online algorithms must make irrevocable decisions for each input item but have no space requirement. In some cases, streaming algorithms are designed so as to make results available at any time during the computation and hence some streaming algorithms can also be viewed as online algorithms as well. Conversely, any online algorithm that restricts itself to $\tilde{O}(n)$ space can be considered as a streaming algorithm.

Priority Model. In the priority model [8], an input to the algorithm is represented as a set of items coming from some universe. The universe of items in the MBM problem is the set of all pairs (online node, its neighborhood). The algorithm orders the entire universe by defining a priority function mapping each possible input item to a real number. Then, the adversary picks an instance G and reveals the online nodes in increasing order as specified by the priority function value. In a fixed-order algorithm, there is one initial ordering of the items whereas in an adaptive-order algorithm, the ordering can be redefined in each iteration. The algorithm makes an irrevocable decision about each input item. This captures many offline greedy algorithms that make a single pass over input items. Many problems have been studied in the priority model [1,5–7,12,33,37]. In a *fully randomized priority algorithm* [1] both the ordering of the input items and the decisions for each item are randomized. When only the decisions are randomized (and the ordering is deterministic) we will simply say *randomized priority algorithm.* With regards to maximum matching, Aronson et al. [2] proved that an algorithm that picks a random vertex and matches it to a random available neighbor (if it exists) achieves approximation ratio $1/2 + \epsilon$ for some $\epsilon > 0$ in general graphs. Besser and Poloczek [5] show that the algorithm that picks a random vertex of minimum degree and matches it to a randomly selected neighbor cannot improve upon the 1/2 approximation ratio (with high probability) even for bipartite graphs. Pena and Borodin [32] show that no deterministic priority algorithm can achieve approximation ratio better than 1/2. See [4,32] with respect to the the difficulty of proving inapproximation results for randomized priority algorithms.

Advice Model. Dürr et al. [15] studied the online MBM problem in the adversarial (tape) *advice model.* In the most unrestricted advice setting, the advice bits are set by an all-powerful oracle. Dürr et al. show that $\Theta_\epsilon(n)$ advice bits are necessary and sufficient to guarantee approximation ratio $1 - \epsilon$ for MBM. They also show that $O(\log n)$ advice bits are sufficient for a deterministic advice algorithm to guarantee a $1 - 1/e$ approximation ratio. Construction of the $O(\log n)$ advice bits is based on examining the behavior of the Ranking algorithm on all $n!$ possible random strings for a given input of length n, which requires exponential time. It is not known if there is an efficient way to construct $O(\log n)$ advice bits. One may put computational or information-theoretic restrictions on the advice string, and ask what approximation ratios are achievable by online algorithms with restricted advice. Beyond their theoretical value, advice algorithms also give rise to classes of conceptually simple offline algorithms if the advice string is restricted to be efficiently computable. The Dürr et al. [15] deterministic advice algorithm Category-Advice achieves approximation ratio 3/5 using an n-bit advice string, where the advice string itself is computable by an online algorithm. This algorithm can be viewed as a 2-pass online algorithm.

Our Online Multipass Results. We generalize the Category-Advice algorithm to a k-pass Category-Advice algorithm for $k \geq 1$. For each $k \geq 1$, we prove that the exact approximation ratio of k-pass Category-Advice algorithm is F_{2k}/F_{2k+1}, where F_n is the nth Fibonacci number. Our bounds show that the analysis of Dürr et al. for the 2-PASS CATEGORY-ADVICE algorithm was tight. Our result immediately implies that the approximation ratio of k-PASS CATEGORY-ADVICE converges to the inverse of the golden ratio $2/(1+\sqrt{5}) \approx 0.618$ as k goes to infinity.

Our Results for the Known IID Model. A greedy algorithm always matches an online vertex if it has at least one available neighbor. In the known IID model, we can consider greedy algorithms without loss of generality (see Remark 1). Moreover, greedy algorithms satisfying natural consistency conditions achieve approximation ratio at least $1 - 1/e$. Ties may occur in a greedy algorithm when an online node has more than one available neighbor. A good tie-breaking rule can improve the approximation ratio. Algorithms beating the $1 - 1/e$ ratio are known in this model ([3, 10, 20, 22, 29]).[4] They are usually stated as non-greedy algorithms, but using a general conversion they can be turned into greedy algorithms, albeit with somewhat unnatural tie-breaking rules. These algorithms have polynomial time preprocessing step and, thus, are feasible from a theoretical point of view, but might not be feasible from a practical point of view on large inputs. Moreover, we argue that these algorithms are not that "conceptually simple." We study a deterministic greedy online algorithm MINDEGREE using a natural, conceptually simple, and efficient tie-breaking rule. This algorithm is motivated by the offline matching algorithm MINGREEDY [36]. We show that MINDEGREE does not beat the $1-1/e$ approximation achieved by any consistent greedy algorithm.

Our Results for the Priority Model. The RANKING algorithm in the ROM model is an instance of a fully randomized priority algorithm. The ordering of the online vertices is simply a uniform random permutation of the set of adversarially chosen input items. Is there a more "informed" way to deterministically or randomly choose the ordering within the priority framework? A natural idea is to give priority to the online nodes having the smallest degree since intuitively they seem to be the hardest to match if not seen early. Our intuition turns out to be misleading. We show that giving priority to nodes having the smallest degree using some deterministic (or the uniform random tie-breaking) rule cannot match the approximation achieved by a uniform ordering of the online nodes. In contrast to the 2-PASS CATEGORY-ADVICE 3/5 approximation, our analysis can also be used to show that a deterministic two-pass algorithm that computes the degree of the offline vertices in the first pass and then reorders the offline vertices according to non-decreasing degree (breaking ties by the initial ordering) will not achieve an asymptotic approximation ratio better than 1/2.

[4] In fact, it is easy to see that there is an optimal tie-breaking rule that can be computed by dynamic programming. Unfortunately, the size of the dynamic programming table is exponentially large, and moreover, currently it is not known how to analyze such optimal tie-breaking rules.

2 Preliminaries

$G = (U, V, E)$ denotes a bipartite graph where $U, V \subset \mathbb{N}$ form a partition of the vertices, and edges are $E \subseteq U \times V$. We consider the MBM in various online models. U represents the online vertices that are revealed one at a time, and V represents the offline vertices. A vertex u from U is revealed together with all edges incident on it. The online models differ in how vertices in U and their arrivals are chosen — adversarially, sampled from a known distribution, or via a limited adversary.

Let $M \subseteq E$ be some matching in G. For $u \in U$, we write $u \in M$ when there exists $v \in V$ such that $(u, v) \in M$. Similarly, we write $v \in M$ when there exists $u \in U$ such that $(u, v) \in M$. For $u \in M$, we write $M(u)$ to indicate the neighbor of u in M. We write $\mathrm{OPT}(G)$ to stand for an offline optimum matching in G. Given an algorithm ALG, we let $\mathrm{ALG}(G)$ stand for the matching returned by the algorithm ALG on input G. Abusing notation, we will also use $\mathrm{ALG}(G)$ (resp. $\mathrm{OPT}(G)$) to stand for the size of the matching returned by the algorithm on G (resp. by $\mathrm{OPT}(G)$).

Definition 1. *We define* asymptotic approximation ratio *of an algorithm* ALG *as* $\mathrm{AR}(\mathrm{ALG}) = \lim_{\mathbb{E}[\mathrm{OPT}(G)] \to \infty} \inf_G \frac{\mathbb{E}[\mathrm{ALG}(G)]}{\mathbb{E}[\mathrm{OPT}(G)]}$, *where the expectation is over the input distribution and randomness of the algorithm. Approximation ratios in other models (e.g., adversarial input, deterministic algorithms, etc.) are defined analogously.*

An online (or priority) MBM algorithm is greedy if whenever a newly arriving online node has at least one available neighbor the algorithm matches the arrived node to one of its neighbors.

Remark 1. Any online (or priority) algorithm for MBM achieving approximation ratio ρ (in an adversarial or stochastic input setting) can be turned into a *greedy* algorithm achieving approximation ratio $\geq \rho$. Informally, the idea is a simulation of the non-greedy algorithm in which we replace any non-greedy decision by forcing a match while remembering the configuration of the non-greedy algorithm.

In the graphs $G = (U, V, E)$ that we consider, we shall often refer to the so-called "parallel edges." Let $U' = \{u'_1, \ldots, u'_k\} \subseteq U$ and $V' = \{v'_1, \ldots, v'_k\} \subseteq V$ be two distinguished subsets such that for all $i \in [k]$ we have $(u'_i, v'_i) \in E$ (there might be other edges incident on U' and V'). The parallel edges between U' and V' are precisely the edges (u'_i, v'_i).

2.1 The Ranking Algorithm

In the adversarial model, graph G as well as an order π of its online vertices U is chosen by an adversary. Karp, Vazirani, and Vazirani [24] presented RANKING (see Algorithm 1). They showed that $\mathrm{AR}(\text{RANKING}) = 1 - 1/e$ and that no randomized algorithm can do better in the adversarial model. RANKING samples a permutation σ of the offline nodes uniformly at random and runs a greedy algorithm on G breaking ties using σ. That is, when u arrives, if u has many unmatched neighbors v then u is matched with v that minimizes $\sigma(v)$. We write Ranking(π, σ) to denote the matching returned by Ranking given π and σ. When π is clear from the context we will omit it and simply write Ranking(σ).

2.2 Known IID Model with Uniform Probability Distribution

Algorithm 1 The Ranking algorithm.

```
procedure Ranking($G = (U, V, E)$)
    Sample permutation $\sigma : V \to V$
      uniformly at random
    for all $u \in U$ do
        When $u$ arrives, set
        $N(u) = \{\text{unmatched } v \text{ s.th. } (u, v) \in E\}$
        if $N(u) \neq \emptyset$ then
            $v^* = \arg\min_v \{\sigma(v) \mid v \in N(u)\}$
            match $u$ with $v^*$
```

In the known IID model, $G = (U, V, E)$ is a *type graph* and nodes U are referred to as "types." The type graph specifies the distribution from which the actual *instance graph* $\widehat{G} = (\widehat{U}, \widehat{V}, \widehat{E})$ is generated. An instance graph is generated by setting $\widehat{V} = V$, sampling each $\widehat{u} \in \widehat{U}$ IID uniformly from U, and letting $(\widehat{u}, \widehat{v}) \in \widehat{E}$ if and only if the corresponding $(u, v) \in E$. Each $\widehat{u}$ is drawn IID from U with replacement, thus it is possible that the same $u \in U$ appears multiple times in $\widehat{U}$. The type graph is chosen adversarially and is revealed to the algorithm in advance. Nodes $\widehat{u}$ are generated and presented to the algorithm one at a time. The algorithm makes an irrevocable decision on how to match $\widehat{u}$ (if at all). More general versions of this model have been defined and studied, but we won't need them in this paper.

A basic guarantee on the performance of a greedy algorithm in the known IID model holds as long as its tie-breaking rules satisfy natural consistency conditions. These conditions are somewhat technical, so we defer them to the full version of the paper [9]. Goel and Mehta [19] proved that such greedy algorithms achieve approximation at least $1 - 1/e$ in the known IID model.[5]

3 Deterministic Multipass Online Algorithms

To break through the 1/2 barrier of the adversarial online model, Dürr et al. [15] modify the model to allow for a second pass over the input. They give a deterministic 2-pass algorithm, called Category-Advice, that achieves a 3/5 approximation ratio. Category-Advice belongs to the class of Ranking-based algorithms called category algorithms that were introduced in the work of Dürr et al. A category algorithm considers a permutation σ of the offline nodes. Instead of running Ranking directly with σ, a category function $c : V \to \mathbb{Z} \cup \{\pm\infty\}$ is

[5] The Goel and Mehta [19] result is even stronger as it holds for the ROM model.

computed first. The updated permutation σ_c is the unique permutation satisfying the following defining property: for all $v_1, v_2 \in V$, we have $\sigma_c(v_1) < \sigma_c(v_2)$ if and only if $c(v_1) < c(v_2)$ or $(c(v_1) = c(v_2)$ and $\sigma(v_1) < \sigma(v_2))$. Then RANKING($\sigma_c$) is performed with σ_c as the permutation of the offline nodes.

The CATEGORY-ADVICE algorithm starts with an arbitrary permutation σ, e.g., σ could be induced by the names of nodes V. In the first pass, the algorithm runs RANKING with σ. Let M be the matching obtained in the first pass. The category function $c : V \to [2]$ is then defined as follows: $c(v) = 1$ if $v \notin M$ and $c(v) = 2$ otherwise. In the second pass, the CATEGORY-ADVICE algorithm runs RANKING(σ_c). The output of the second run of RANKING is declared as the output of the CATEGORY-ADVICE algorithm. In other words, in the second pass the algorithm gives preference to those vertices that were **not** matched in the first pass. (We observe that the Besser-Poloczek graphs, see the full version of the paper [9], show that the category function $c(v) =$ degree of v will not yield an asymptotic approximation better than $1/2$.)

We present a natural extension of the CATEGORY-ADVICE algorithm to multiple passes, called k-PASS CATEGORY-ADVICE (see Algorithm 2). Let F_n denote the nth Fibonacci number, i.e., $F_1 = F_2 = 1$, and $F_n = F_{n-1} + F_{n-2}$ for $n \geq 3$. For each $k \geq 1$, we prove that the k-PASS CATEGORY-ADVICE algorithm[6] achieves the approximation ratio F_{2k}/F_{2k+1}. Moreover, we show that this is tight, i.e., there exists a family of bipartite graphs, one for each $k \geq 1$, such that the k-PASS CATEGORY-ADVICE algorithm computes a matching that is F_{2k}/F_{2k+1} times the size of the maximum matching. In particular, we show that the analysis of the $3/5$ approximation ratio in [15] is tight. It immediately follows that as k goes to infinity, the approximation guarantee of the k-PASS CATEGORY-ADVICE algorithm converges (quickly) to the inverse of the golden ratio $2/(1+\sqrt{5}) \approx 0.618$. The main theorem of this section is

Theorem 1. *The exact approximation ratio of the k-PASS CATEGORY-ADVICE algorithm is F_{2k}/F_{2k+1}, where F_n is the nth Fibonacci number. Thus, the approximation ratio of the k-PASS CATEGORY-ADVICE algorithms tends to the inverse of the golden ration $2/(1+\sqrt{5}) \approx 0.618$ as k goes to infinity. This holds even when k depends on n arbitrarily. The algorithm can be realized as a k-pass semi-streaming algorithm using $O(n \log k)$ space, which is $O(n)$ for constant k.*

3.1 Positive Result

The pseudocode for k-PASS CATEGORY-ADVICE appears in Algorithm 2. The algorithm is defined iteratively with each iteration corresponding to a new pass. The algorithm initializes σ of the offline nodes to the identity permutation. The algorithm maintains a category function $c : V \to \mathbb{Z} \cup \{\pm\infty\}$. Initially, c is set to

[6] A notable feature of this multi-pass algorithm is that after pass i, the algorithm can deterministically commit to matching a subset of size $\frac{F_{2i}}{F_{2i+1}}|M|$ where M is a maximum matching. This follows from a certain monotonicity property. See the full version of the paper for details [9].

$-\infty$ everywhere. In the ith pass, the algorithm runs RANKING on σ_c. Let M_i be the resulting matching. For each $v \in V$, if $c(v) = -\infty$ and $v \in M_i$ then $c(v)$ is updated to $-i$. The algorithm uses the updated c in the next pass. In words, c records for each vertex v the (negative of the) first pass in which v was matched. In the subsequent pass, the algorithm gives preference to the nodes that were unmatched, followed by nodes that were matched for the first time in the latest round, etc.

Lemma 1. *The k-PASS CATEGORY-ADVICE algorithm achieves approximation ratio F_{2k}/F_{2k+1}.*

Algorithm 2 k-PASS CATEGORY-ADVICE.

```
procedure k-PASS CATEGORY ADVICE
    $\sigma \leftarrow$ the identity permutation of $V$
    initialize array $c$ of size $V$ with $-\infty$
    for $i$ from 1 to $k$ do
        Run pass $i$: $M_i \leftarrow$RANKING$(\sigma_c)$
        for $v \in V$ do
            if $c(v) = -\infty$ and $v \in M_i$ then
                $c(v) \leftarrow -i$
    return $M_k$
```

Proof (by induction on k). Base case: $k = 1$. The 1-PASS CATEGORY-ADVICE algorithm is the simple deterministic greedy algorithm, which achieves a $1/2 = F_2/F_3$ approximation ratio.

Inductive step. Consider the $(k+1)$-PASS CATEGORY-ADVICE algorithm running on $G = (U, V, E)$. WLOG, we may assume that $|U| = |V|$ and G has a perfect matching (see the full version of the paper [9]). Let $U_1 \subseteq U$ be the set of nodes matched in the first pass of the algorithm. Let $V_1 \subseteq V$ be the nodes that U_1 nodes are matched to. Define $U_2 = U \setminus U_1$ and $V_2 = V \setminus V_1$. Note that there are no edges between U_2 and V_2; otherwise some node of U_2 would have been matched to some node of V_2 in the first round. Let M_i be the matching found by the $(k+1)$-PASS CATEGORY-ADVICE algorithm in round i, where $i \in [k+1]$. Also define $M_{11} = M_{k+1} \cap U_1 \times V_1$, $M_{12} = M_{k+1} \cap U_1 \times V_2$, and $M_{21} = M_{k+1} \cap U_2 \times V_1$. We are interested in bounding $|M_{k+1}| = |M_{11}| + |M_{12}| + |M_{21}|$. See Figure 1.

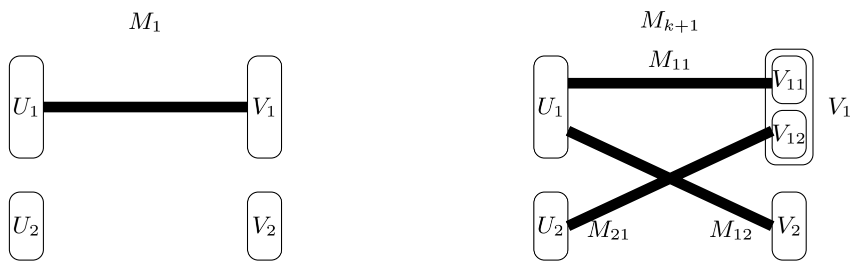

Fig. 1. G is the input graph. On the left we show the matching constructed in the first pass. On the right we show the matching constructed in the $k+1$st pass.

After the first round, nodes in U_1 prefer nodes from V_2 to those from V_1. Moreover, nodes in V_2 are only connected to nodes in U_1 and there is a perfect matching between V_2 and a subset of U_1. Thus, the matching constructed between U_1 and V_2 in the next k passes is the same as if we ran k-PASS CATEGORY-ADVICE algorithm on the subgraph of G induced by $U_1 \cup V_2$. This implies that $|M_{12}| \geq (F_{2k}/F_{2k+1})|V_2| = (F_{2k}/F_{2k+1})|U_2|$.

By the monotonicity property (see the full version of the paper [9]), in the $k + 1$st pass, all nodes from U_1 that were not matched with V_2 will be matched with some nodes in V_1, i.e., $|U_1| = |M_{12}| + |M_{11}|$. Let V_{11} be such a set, and let $V_{12} = V_1 \setminus V_{11}$. To lower bound $|M_{21}|$, we first lower bound the size of a maximum matching between U_2 and V_{12}. Since U_2 is only connected to V_1 and since there is a perfect matching, a maximum matching between U_2 and V_1 is of size $|U_2|$. Thus, the size of a maximum matching between U_2 and V_{12} is at least $|U_2| - |V_{11}|$. Also, observe that $|V_{11}| = |V_1| - |V_{12}|$ and $|V_{12}| = |M_{12}|$. Therefore, the size of a maximum matching between U_2 and V_{12} is at least $|U_2| - (|V_1| - |M_{12}|) = |U_2| - |U_1| + |M_{12}|$ (note that $|U_1| = |V_1|$). Finally in the last round, the algorithm constructs a maximal matching between U_2 and V_{12} guaranteeing that $|M_{21}| \geq (1/2)(|U_2| - |U_1| + |M_{12}|)$. Putting it all together, we obtain $|M_{k+1}| = |M_{11}| + |M_{12}| + |M_{21}| = |U_1| - |M_{12}| + |M_{12}| + |M_{21}| \geq |U_1| + \frac{1}{2}(|U_2| - |U_1| + |M_{12}|) \geq \frac{1}{2}\left(|U_2| + |U_1| + \frac{F_{2k}}{F_{2k+1}}|U_2|\right) = \frac{1}{2}\left(n + \frac{F_{2k}}{F_{2k+1}}(n - |M_1|)\right) = \left(\frac{1}{2} + \frac{F_{2k}}{F_{2k+1}}\right)n - \frac{F_{2k}}{2F_{2k+1}}|M_1|$.
By the monotonicity property we also have that $|M_{k+1}| \geq |M_1|$. Thus, we derive $|M_{k+1}| \geq \max\left\{|M_1|, \left(\frac{1}{2} + \frac{F_{2k}}{F_{2k+1}}\right)n - \frac{F_{2k}}{2F_{2k+1}}|M_1|\right\}$. This implies that $|M_{k+1}| \geq \frac{1/2 + F_{2k}/F_{2k+1}}{1 + F_{2k}/(2F_{2k+1})}n = \frac{F_{2k+1} + F_{2k}}{2F_{2k+1} + F_{2k}}n = \frac{F_{2(k+1)}}{F_{2(k+1)+1}}n$.

3.2 Negative Result

In this subsection, we construct a family of bipartite graphs $(G_k)_{k=1}^{\infty}$ with the following properties[7]: (1) G_k has exactly F_{2k+1} online nodes and F_{2k+1} offline nodes, (2) G_k has a perfect matching, (3) the k-PASS CATEGORY-ADVICE algorithm finds a matching of size F_{2k}, (4) for all $k' > k$, the k'-PASS CATEGORY-ADVICE algorithm finds a matching of size $F_{2k} + 1$.

This family of graphs shows that our analysis of k-PASS CATEGORY-ADVICE is tight. The construction of the G_k is recursive. The base case is given by G_1, which is depicted in Figure 2. The online nodes are shown on the left whereas the offline nodes are on the right. The online nodes always arrive in the order shown on the diagram from top to bottom, and the initial permutation σ of the offline nodes is also given by the top to bottom ordering of the offline nodes on the diagram.

[7] The last property allows us to conclude that the approximation ratio of k-PASS CATEGORY-ADVICE converges to the inverse of the golden ratio even when k is allowed to depend on n arbitrarily.

In the recursive step, we construct G_{k+1} from G_k. The online nodes U of G_{k+1} are partitioned into three disjoint sets $U = U_1 \cup U_2 \cup U_3$ such that $|U_1| = |U_3| = F_{2k+1}$ and $|U_2| = F_{2k}$. Similarly, the offline nodes V of G_{k+1} are partitioned into three disjoint sets $V = V_1 \cup V_2 \cup V_3$ such that $|V_1| = |V_3| = F_{2k+1}$ and $|V_2| = F_{2k}$. There is a copy of G_k between U_1 and V_3. U_2 and V_2 are connected by parallel edges. There is a complete bipartite graph between U_1 and V_1, as well as between U_2 and V_1. Finally, U_3 is connected to V_1 by parallel edges. The construction is depicted in Figure 3.

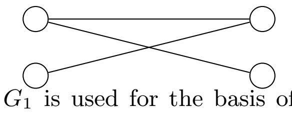
Fig. 2. G_1 is used for the basis of the induction.

Lemma 2. *Properties (1-4) mentioned at the beginning of this subsection hold for the G_k.*

Proof (by induction on k). Base case: $k = 1$. G_1 has 2 online and 2 offline nodes and $F_3 = 2$. Clearly, G_1 has a perfect matching. The 1-PASS CATEGORY-ADVICE algorithm is the regular greedy algorithm, which returns a matching of size 1 on G_1, and $F_2 = 1$. Lastly, 2-PASS CATEGORY-ADVICE finds a matching of size $F_2 + 1 = 2$ and adding more passes does not change anything.

Inductive step. Assume that properties (1-4) hold for G_k. The number of online vertices of G_{k+1} is equal to the number of offline vertices and is equal to $F_{2k+1} + F_{2k} + F_{2k+1} = F_{2k+2} + F_{2k+1} = F_{2k+3} = F_{2(k+1)+1}$. By inductive assumption, G_k has a perfect matching. Therefore, U_1 vertices can be matched with V_3 via the perfect matching given by G_k. In addition, U_2 can be matched with V_2 by parallel edges, and U_3 can be matched with V_1 by parallel edges as well. Thus, G_{k+1} has a perfect matching. Thus, we proved properties 1 and 2. To prove the 3rd property, observe that in the first pass, the $(k+1)$-PASS CATEGORY-ADVICE algorithm matches U_1 and V_1 by parallel edges, U_2 with V_2 by parallel edges, and leaves U_3 and V_3 unmatched. Since V_3 is only connected to the nodes U_1, in the next k passes the behavior of the algorithm between U_1 and V_3 nodes is exactly that of the k-PASS CATEGORY-ADVICE algorithm. Therefore, by the inductive assumption, the algorithm is going to match exactly F_{2k} nodes from U_1 with the nodes in

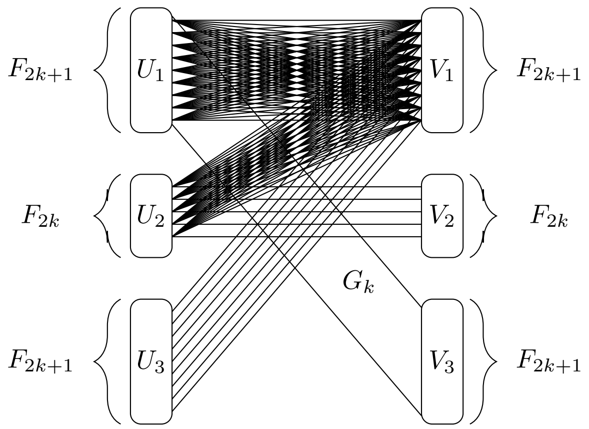

Fig. 3. G_{k+1} shows the inductive construction.

V_3. The remaining $F_{2k+1} - F_{2k}$ nodes from U_1 will be matched to the nodes in V_1 (since those are the only neighbors of U_1 nodes besides the nodes from V_3). The nodes from U_2 in all passes behave the same way – they prefer V_1 nodes to V_2 nodes. Thus, since V_1 will have F_{2k} nodes unmatched after processing all nodes of U_1 in the last round, all of U_2 nodes will be matched to V_1 in the last round. This implies that after processing U_1 and U_2 in the last round, all of V_1 nodes are matched. Therefore, none of U_3 nodes can be matched. Thus, the matching found by the $(k+1)$-PASS CATEGORY-ADVICE algorithm on G_{k+1} is of size $|U_1| + |U_2| = F_{2k+1} + F_{2k} = F_{2k+2} = F_{2(k+1)}$. The last property is proved similarly.

4 MinDegree Algorithm in the Known IID Model

Our algorithm is MINDEGREE (see Algorithm 3). The motivation for studying this algorithm is as follows. It is easy to see that in the adversarial setting we can take any greedy algorithm, modify the tie-breaking rule to always give more preference to the offline nodes of degree 1, and this will not decrease the approximation ratio of the algorithm. Generalizing this, we conclude that online vertices should give preference to the offline vertices of smaller degrees. The problem is that in the adversarial setting, degrees of the offline nodes are not known a priori. However, in the known IID setting we can estimate the degrees of the offline vertices from the type graph. This is precisely what MINDEGREE formalizes. The algorithm is given a type graph $G = (U, V, E)$ as input. It keeps track of a set S of currently "active", i.e., not yet matched, offline nodes. When a new node $\widetilde{u}$ arrives, it is matched to its active neighbor of minimum degree in the type graph.

Algorithm 3 The MINDEGREE algorithm.

```
procedure MINDEGREE($G = (U, V, E)$)
    Let $S = V$ be the set of active offline nodes
    repeat
        Let $\widehat{u}$ denote the arriving online node
        Let $N(\widehat{u}) = \{v \in S \mid (u, v) \in E\}$
        if $N(\widehat{u}) \neq \emptyset$ then
            Let $v = \arg\min_{v \in N(\widehat{u})} \deg(v)$
            Match $\widehat{u}$ with $v$
            Remove $v$ from $S$
    until all online nodes have been processed
```

Remark 2. Our algorithm does not fully break ties, i.e., MINDEGREE takes *some* neighbor of currently minimum degree. In practice, it means that ties are broken in some arbitrary way, e.g., by names of vertices. In our theoretical analysis, this means that the adversary is more powerful, as it can decide how the algorithm breaks these ties.

MINDEGREE is a conceptually simple and promising algorithm in the known IID setting. Indeed, a version of MINDEGREE called MINGREEDY has been extensively studied in the offline setting (see Besser and Poloczek [5] and references therein). Unfortunately, in spite of having excellent empirical performance as well as excellent performance under various random input models, MinGreedy has a bad worst-case approximation ratio of 1/2 in the offline adversarial setting [5]. As far as we know, this algorithm has not been analyzed in the known IID model. We obtain a result that, in spirit, is similar to the offline result, but the proof is different. Namely, we show that MinDegree cannot achieve an approximation ratio better than $1-1/e$, which is guaranteed by any consistent greedy algorithm in the known IID model. See the full version of the paper [9] for the proof of the following result.

Theorem 2.

$$\text{AR}(\text{MinDegree}) = 1 - \frac{1}{e}$$

Th negative result holds no matter which rule is used to break (remaining) ties in MinDegree*.*

5 MinRanking – A Hybrid Algorithm in the Priority Model

We propose a conceptually simple greedy algorithm for MBM. Our algorithm is a natural hybrid between two well-known algorithms—Ranking and MinGreedy. We have already encountered MinGreedy in this paper (see Section 1 and Section 4). MinGreedy is an offline algorithm that selects a random vertex of minimum degree in the input graph and matches it with a random neighbor, removes the matched vertices, and proceeds. In a natural adaptation of MinGreedy to bipartite graphs $G = (U, V, E)$, the algorithm picks a random node of minimum degree from U and matches it to a random neighbor from V. Observe that this algorithm can be realized as a fully randomized priority algorithm, where the ordering of the input items is by increasing degree with ties broken randomly. See the full version of the paper [9] for the pseudocode.

Karp, Vazirani, Vazirani [24] exhibited a family of graphs, on which Ranking gets its worst approximation ratio $1-1/e$. MinGreedy finds a perfect matching on these graphs. Thus, it is natural to consider the performance of an algorithm that combines Ranking and MinGreedy. This is our proposed MinRanking algorithm (see Algorithm 4). In MinRanking, a random permutation π of vertices V is initially sampled. Then, nodes in U are processed in the increasing order of their current degrees (cur deg) with ties broken randomly. When a node u is processed, it is matched with its neighbor appearing earliest in the ordering π.

MinRanking modifies MinGreedy in the same way that the online Ranking algorithm modifies the seemingly more natural online randomized algorithm that simply matches an online vertex to an available neighbor uniformly at random which surprisingly was shown to be (asymptotically) a 1/2 approximation. So it is hopeful that MinRanking can improve upon MinGreedy. Like MinGreedy, our algorithm can be implemented and analyzed in the fully randomized adaptive priority model [8]. Since our algorithm is a generalization of Ranking, its asymptotic approximation ratio is at least $1 - 1/e \approx 0.632$. We show that the asymptotic approximation ratio of this algorithm is at most $1/2 + 1/(2e) \approx 0.684$, as witnessed by the construction of Besser and Poloczek [5].

Theorem 3.

$$1 - \frac{1}{e} \leq \mathrm{AR}(\textsc{MinRanking}) \leq \frac{1}{2} + \frac{1}{2e}.$$

This negative result shows that MinRanking falls short of the known bound for Ranking in the ROM model, where it achieves approximation ratio 0.696 [28]. From our result it follows that a deterministic ordering of the online nodes by non-decreasing degree (breaking ties by the given initial labeling of those nodes) will also fall short. That is (similar to the result in [32] for deterministic decisions), a naive randomized ordering can be better than a seemingly informed deterministic ordering. See the full version of the paper [9] for more details.

Algorithm 4 The MinRanking algorithm.

```
procedure MinRanking($G = (U, V, E)$)
    Pick a permutation $\pi : V \to V$ at random
    repeat
        Let $d = \min\{\mathrm{cur\,deg}(i) \mid i \in U\}$
        Let $S = \{i \in U \mid \mathrm{cur\,deg}(i) = d\}$
        Pick $i \in S$ uniformly at random
        Let $N(i)$ be the set of neighbors of $i$
        if $N(i) = \emptyset$ then
            $i$ remains unmatched
            Delete $i$ from $G$
        else
            $j = \arg\min_k \{\pi(k) \mid k \in N(i)\}$
            Match $i$ with $j$
            Delete $i$ and $j$ from $G$
            Update cur deg
    until $U = \emptyset$
```

6 Conclusion and Open Problems

We have considered a number of "online-based" algorithms for the MBM problem. We believe that the algorithms considered in this paper all pass the intuitive "you know it when you see it" standard for conceptually simple algorithms. In particular, these algorithms take linear time in the number of edges and are very easy to implement. Even given the restricted nature of these algorithms, it is a challenge to understand their performance.

Our results for the MBM, in conjunction with the results in Poloczek et al. [34] for MaxSat, show both the promise and limitations of conceptually simple algorithms. Many open problems are suggested by this work. Clearly, any problem studied in the competitive online literature can be considered within the expanded framework of conceptually simple algorithms. For what problems is there a general method for de-randomizing online algorithms? Is there a precise algorithmic model that lends itself to analysis and captures multi-pass algorithms? And in addition to worst-case and stochastic analysis, how would any of the conceptually simple MBM algorithms perform "in practice?"

References

1. Spyros Angelopoulos and Allan Borodin. On the power of priority algorithms for facility location and set cover. In *Proc. of APPROX*, pages 26–39, 2002.
2. Jonathan Aronson, Martin Dyer, Alan Frieze, and Stephen Suen. Randomized greedy matching. II. *Random Struct. Algorithms*, 6(1):55–73, 1995.
3. Bahman Bahmani and Michael Kapralov. Improved bounds for online stochastic matching. In *Proc. of ESA*, pages 170–181, 2010.
4. Bert Besser and Matthias Poloczek. Erratum to: Greedy matching: Guarantees and limitations. *Algorithmica*, pages 1–4, 2017.
5. Bert Besser and Matthias Poloczek. Greedy matching: Guarantees and limitations. *Algorithmica*, 77(1):201–234, 2017.
6. Allan Borodin, Joan Boyar, and Kim S. Larsen. *Priority Algorithms for Graph Optimization Problems*, pages 126–139. Springer Berlin Heidelberg, 2005.
7. Allan Borodin, Ioana Ivan, Yuli Ye, and Bryce Zimny. On sum coloring and sum multi-coloring for restricted families of graphs. *Theoretical Computer Science*, 418:1 – 13, 2012.
8. Allan Borodin, Morten N. Nielsen, and Charles Rackoff. (incremental) priority algorithms. *Algorithmica*, 37(4):295–326, 2003.
9. Allan Borodin, Denis Pankratov, and Amirali Salehi-Abari. On conceptually simple algorithms for variants of online bipartite matching. *CoRR*, abs/1706.09966, 2017.
10. Brian Brubach, Karthik Abinav Sankararaman, Aravind Srinivasan, and Pan Xu. New Algorithms, Better Bounds, and a Novel Model for Online Stochastic Matching. In *Proc. of ESA*, pages 24:1–24:16, 2016.
11. Barun Chandra and Magnús M. Halldórsson. Greedy local improvement and weighted set packing approximation. *J. Algorithms*, 39(2):223–240, 2001.
12. Sashka Davis and Russell Impagliazzo. Models of greedy algorithms for graph problems. *Algorithmica*, 54(3):269–317, 2009.
13. Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113, 2008.
14. Ran Duan and Seth Pettie. Linear-time approximation for maximum weight matching. *J. ACM*, 61(1):1:1–1:23, 2014.
15. Christoph Dürr, Christian Konrad, and Marc Renault. On the Power of Advice and Randomization for Online Bipartite Matching. In *Proc. of ESA*, pages 37:1–37:16, 2016.
16. Sebastian Eggert, Lasse Kliemann, Peter Munstermann, and Anand Srivastav. Bipartite matching in the semi-streaming model. *Algorithmica*, 63(1-2):490–508, 2012.
17. Jon Feldman, Aranyak Mehta, Vahab Mirrokni, and S. Muthukrishnan. Online stochastic matching: Beating 1-1/e. In *Proc. of FOCS*, pages 117–126, 2009.
18. Ashish Goel, Michael Kapralov, and Sanjeev Khanna. On the communication and streaming complexity of maximum bipartite matching. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 468–485, 2012.
19. Gagan Goel and Aranyak Mehta. Online budgeted matching in random input models with applications to adwords. In *Proc. of SODA*, pages 982–991, 2008.
20. Bernhard Haeupler, Vahab S. Mirrokni, and Morteza Zadimoghaddam. Online stochastic weighted matching: Improved approximation algorithms. In *Proc. of WINE*, pages 170–181, 2011.

21. John E. Hopcroft and Richard M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM J. on Comput.*, 2(4):225–231, 1973.
22. Patrick Jaillet and Xin Lu. Online stochastic matching: New algorithms with better bounds. *Mathematics of Operations Research*, 39(3):624–646, 2014.
23. Michael Kapralov. Better bounds for matchings in the streaming model. In *Proc. of SODA*, pages 1679–1697, 2013.
24. R. M. Karp, U. V. Vazirani, and V. V. Vazirani. An optimal algorithm for on-line bipartite matching. In *Proc. of STOC*, pages 352–358, 1990.
25. Brendan Lucier and Vasilis Syrgkanis. Greedy algorithms make efficient mechanisms. In *Proc. of Conference on Economics and Computation, EC*, pages 221–238, 2015.
26. A. Madry. Navigating central path with electrical flows: From flows to matchings, and back. In *Proc. of FOCS*, pages 253–262, 2013.
27. A. Madry. Computing maximum flow with augmenting electrical flows. In *Proc. of FOCS)*, pages 593–602, 2016.
28. Mohammad Mahdian and Qiqi Yan. Online bipartite matching with random arrivals: An approach based on strongly factor-revealing LPs. In *Proc. of STOC*, pages 597–606, 2011.
29. Vahideh H. Manshadi, Shayan Oveis Gharan, and Amin Saberi. Online stochastic matching: Online actions based on offline statistics. In *Proc. of SODA*, pages 1285–1294, 2011.
30. Andrew McGregor. Graph sketching and streaming: New approaches for analyzing massive graphs. In *Proc. of Intern. Comput. Sci. Symp. in Russia, CSR*, pages 20–24, 2017.
31. M. Mucha and P. Sankowski. Maximum matchings via gaussian elimination. In *Proc. of FOCS*, pages 248–255, 2004.
32. Nicolas Pena and Allan Borodin. On the limitations of deterministic derandomizations for online bipartite matching and max-sat. *CoRR*, abs/1608.03182, 2016.
33. Matthias Poloczek. *Bounds on Greedy Algorithms for MAX SAT*, pages 37–48. 2011.
34. Matthias Poloczek, Georg Schnitger, David P. Williamson, and Anke Van Zuylen. Greedy algorithms for the maximum satisfiability problem: Simple algorithms and inapproximability bounds. *SICOMP*. Accepted for publication.
35. Matthias Poloczek and David P. Williamson. An experimental evaluation of fast approximation algorithms for the maximum satisfiability problem. In *Proc. of Intern. Symp. on Experimental Algorithms, SEA*, pages 246–261, 2016.
36. Gottfried Tinhofer. A probabilistic analysis of some greedy cardinality matching algorithms. *Annals of Operations Research*, 1(3):239–254, 1984.
37. Yuli Ye and Allan Borodin. Priority algorithms for the subset-sum problem. *J. Comb. Optim.*, 16(3):198–228, 2008.