Noname manuscript No.
(will be inserted by the editor)

On Conceptually Simple Algorithms for Variants of Online Bipartite Matching

Allan Borodin · Denis Pankratov[⋆] · Amirali Salehi-Abari

Received: date / Accepted: date

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 in this model is 1/2, which is achieved by any greedy algorithm. Dürr et al. [23] presented a 2-pass algorithm CATEGORY-ADVICE with 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 quickly to the inverse of the golden ratio $2/(1+\sqrt{5}) \approx 0.618$ as k goes to infinity.

We then consider a natural adaptation of a well-known offline MINGREEDY algorithm to the online stochastic IID model, which we call MINDEGREE. In spite of excellent empirical performance of MINGREEDY, it was recently shown to have approximation ratio 1/2 in the adversarial offline setting — the approximation ratio achieved by any greedy algorithm. Our result in the online known IID model is, in spirit, similar to the offline result, but the proof is different. 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.

Finally, following the work in Besser and Poloczek [7], we depart from an adversarial or stochastic ordering and investigate a natural randomized algorithm (MINRANKING) in the priority model. Although the priority model allows the algorithm to choose the input ordering in a general but well defined way, this natural algorithm cannot obtain the approximation of the RANKING algorithm in the ROM model.

Keywords Conceptually simple algorithms · Online algorithms · Priority algorithms · Bipartite matching · Greedy algorithms

1 Introduction

Maximum bipartite matching (MBM) is a classic extensively studied 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. In 1931, König and Egerváry independently gave a characterization of MBM in terms of the minimum vertex cover. The celebrated Hopcroft-Karp algorithm [33] running in time $O(m\sqrt{n})$ was discovered in

⋆ The work was done while the author was a postdoc at the University of Toronto.

Research is supported by NSERC.

A. Borodin
University of Toronto, Department of Computer Science
Tel.: +1-416-978-6416
E-mail: bor@cs.toronto.edu

D. Pankratov
Concordia University, Department of Computer Science and Software Engineering
Tel.: +1-514-848-2424 ext 7309
E-mail: denis.pankratov@concordia.ca

A. Salehi-Abari
University of Ontario Institute of Technology, Faculty of Business and IT
Tel.: +1-905-721-8668 ext 5544 E-mail: abari@uoit.ca

1973 where n is the number of vertices. The first improvement in the regime of relatively sparse graphs came forty years later, in 2013, when Madry [42] 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 [50] describe an algorithm running in time $O(n^\omega)$, where $\omega \leq 2.373$ is the matrix multiplication constant. We refer the interested reader to [43] and [22] 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 [22]. Arguably, such algorithms are not that conceptually simple and require a reasonable understanding of the problem.

Conceptually Simple Algorithms. Our focus in this paper is on "conceptually simple algorithms" and, in particular, on such algorithms with regard to variants of the online bipartite matching 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 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[1] input models). In all of 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 much 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 simple algorithms for MaxSat [55], greedy algorithms for matching in general graphs [44], degree-based matching algorthms for input-queued switches [34], and greedy based algorithms for bipartite matching [12].

However, existence of efficient FPTAS algorithms running in time $O(m/\epsilon)$ for bipartite matching still begs the question as to why we should restrict ourselves to conceptually simple algorithms. 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 substantially 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 [18] and Poloczek and Williamson [55]. Moreover, and this can be even more critical in many settings, conceptually simple algorithms are relatively easy to implement, verify, and modify with relatively little knowledge about the problem domain. Indeed, conceptual simplicity is arguably the main reason for the use of simple mechanisms in auctions (see, for example, Lucier and Syrgkanis [41]) and the success of MapReduce in distributed parallel applications as introduced by Dean and Ghemawat [20].

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.[23]. In this regard we are also motivated by the Poloczek et al. [53] two-pass algorithm for MaxSat. The Dürr et al. two-pass algorithm and our extension to a k-pass online bipartite algorithm can also be viewed as an $O(n \log n)$ bit space semi-streaming algorithm in the vertex input model. We can also view these multi-pass algorithms as de-randomizations of randomized online algorithms. The second departure is that of priority algorithms [13], a model for greedy and more generally myopic algorithms 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 decision (e.g., as in [26]) and parallel executions of online algorithms (e.g., as in [32,17,51]).

We now outline the computational models relevant to our focus on simple algorithms, and then highlight our results. In Appendix A, we elaborate on the discussion of these specific algorithmic models.

Adversarial Online Model. In 1990, Karp, Vazirani, and Vazirani [38] initiated the study of MBM in the online setting. In this setting, nodes V are the offline nodes known to an algorithm beforehand, and nodes U arrive online in some adversarial order. When a node in U arrives, all its neighbors in V are revealed simultaneously. An online algorithm is required to make an irrevocable decision with regard to which neighbor (if any) the arrived node is matched.

Online Stochastic Models. Motivated by practical considerations, Feldman et al. [28] introduced the known IID distributional model for MBM. Feldman et al. model statistics about the upcoming queries (online nodes) 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 and are presented to an algorithm

[1] IID stands for "independent identically distributed."

making irrevocable decisions. The algorithm knows the type graph and p in advance. For a thorough treatment of the pre-2012 results on this and related models and problems (e.g., Adwords, Display, Welfare Maximization), we refer the reader to the excellent survey by Mehta [48]. Rapid growth of the online advertising industry led to such an abundance of related online models and problems. Online advertising platforms essentially solve various flavors of MBM. Online MBM is thus an important example of a problem that is at the intersection of practice and theory.

Semi-streaming Model. One-pass and multi-pass streaming algorithms are important algorithmic models motivated by the necessity to process extremely large data streams where the amount of data may preclude storing all of the input and hence small space bounds are imposed throughout the computation. The usual assumption is that the data stream is created by an adversary but other input streams (e.g., distributional) are also possible. 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 usually requires $\Omega(n \log n)$ space and hence a realistic goal [2] is to maintain $\tilde{O}(n)$ space rather than space $O(m)$.

Priority Model. The priority model [13] captures the notion of greedy-like algorithms. In this setting, an input to the algorithm is represented as a set of input items coming from some (possibly infinite) universe. The algorithm first commits to an ordering of the entire universe.[3] Then, the adversary picks an instance, which is a finite subset of the universe, and reveals the online items in the order specified by the algorithm. The algorithm is still required to make an irrevocable decision about a newly arriving item.

Advice Model. Dürr et al. in [23] studied the online MBM problem in the adversarial (tape) *advice model.* (See [14] for a survey of online algorithms with advice.) Advice can be viewed as a generalization of randomness. A randomized algorithm is given random bits (independent of the input instance), whereas in the most unrestricted advice setting, the advice bits are set by an all-powerful oracle with knowledge of the entire input.

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. Not only is the advice framework of theoretical value, but it also gives rise to classes of conceptually simple offline algorithms if the advice string is restricted to be efficiently computable. Dürr et al. [23] present a deterministic advice algorithm CATEGORY-ADVICE achieving approximation ratio 3/5 with an n-bit advice string, where the advice string itself is computable by an online algorithm.

1.1 Summary of Our Results

In this subsection, we briefly describe our results on conceptually simple algorithms under the input models discussed above. For convenience, the results are summarized in Table 1.

Algorithm	Model	Positive Result	Negative Result
k-PASS CATEGORY ADVICE	Online Multi-Pass	F_{2k}/F_{2k+1}	F_{2k}/F_{2k+1}
MINDEGREE	Known IID	$1-1/e$	$1-1/e$
MINRANKING	Priority	$1-1/e$	$1/2+1/(2e)$

Table 1 Summary of results in this paper. The positive results for MINDEGREE and MINRANKING trivially follow from prior work. The online Multi-Pass model is not formally defined, although formalizations of this model based on advice model are possible.

Our Online Multi-pass Results. We generalize the CATEGORY-ADVICE algorithm ([23]) to a k-PASS CATEGORY-ADVICE algorithm for $k \geq 1$. For each $k \geq 1$, we prove that the exact approximation ratio[4] 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

[2] Since the space for specifying the output of a streaming algorithm is included in the overall space requirement, any streaming algorithm for say bipartite matching requires at least $\Omega(n \log n)$ space complexity when measured in bits or $\Omega(n)$ complexity when space is measured in words of memory.

[3] In practice, the ordering is determined by a priority function that maps each possible input item into a real valued priority and then the ordering is determined by these priority values.

[4] Our positive result says that the k-PASS CATEGORY-ADVICE algorithm finds a matching of size $\lceil (F_{2k}/F_{2k+1})n \rceil$ in a graph with maximum matching of size n. However, for ease of presentation we shall omit ceilings and floors.

to the inverse of the golden ratio $2/(1+\sqrt{5}) \approx 0.618$ as k goes to infinity. In contrast to the 2-pass Category-Advice 3/5 approximation, we observe that a deterministic two-pass algorithm that computes the degrees 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. Lastly, we give a generalized definition of a 2-pass category-based advice algorithm and conjecture that no such algorithm (even randomized) with $O(1)$ categories can achieve an approximation ratio better than $1 - 1/e$.

Our Results for the Known IID Model. A greedy algorithm always matches an online vertex if it has at least one available neighbor. As observed in Pena and Borodin [51], every online algorithm for bipartite matching can be converted into a greedy online algorithm without hurting its approximation ratio. The greedy property does not specify how to break ties when several neighbors of an online vertex are available. Goel and Mehta [30] proved that most greedy algorithms achieve at least a $1 - 1/e$ approximation ratio in the known IID model[5]. This performance is guaranteed no matter what tie-breaking rule is used as long as the greedy algorithm is "consistent"; see Definition 4 in Appendix B for a precise definition of a consistent greedy algorithm. Algorithms achieving better approximation ratios are known. 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 of the type graph and, thus, are feasible from a theoretical point of view. The preprocessing step, although polynomial time, might not be feasible from a practical point of view on large type graphs. Moreover, we argue that these algorithms are not that "conceptually simple." At present, the research literature on conceptually simple tie-breaking rules for greedy algorithms in the known IID model is scarce. We introduce and study a deterministic greedy online algorithm arising out of a natural, conceptually simple, and efficient tie-breaking rule — MinDegree. This algorithm is motivated by a well-known offline matching algorithm — MinGreedy [56]. We show that even for the known IID model with a uniform distribution, MinDegree does not beat the $1 - 1/e$ approximation achieved by any greedy algorithm (using consistent tie-breaking).

Our Results for the Priority Model. When we consider the randomized Ranking algorithm in the ROM model, we have an instance of a priority algorithm where both the order of input arrivals and the decisions made for each online input are randomized. This ordering is simply a uniform random permutation of the set of adversarial 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 would seem to be the hardest to match if not seen early. Using the Ranking algorithm to match online nodes, 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.

Organization. The rest of this paper is organized as follows. Section 2 introduces key definitions, notation, and a statement of the Ranking algorithm. In Section 3 we describe the k-pass Category-Advice algorithm and prove the exact approximation ratio for each $k \geq 1$, giving matching upper and lower bounds. In Section 4, we provide an analysis of MinDegree in the known IID model. In Section 5, we describe the MinRanking algorithm and provide analysis of the family of graphs due to Besser and Poloczek [7]. We conclude and present some open problems in Section 6.

2 Preliminaries

We are given a bipartite graph $G = (U, V, E)$ where U and V form a partition of the vertices, and edges are $E \subseteq U \times V$. Names of the vertices are natural numbers. We consider the maximum bipartite matching problem in various online models. We think of U as the online vertices that are revealed to a given algorithm one at a time, and V as the offline vertices. When a vertex u from U is revealed, all edges incident on u are also revealed simultaneously. The online models differ in how vertices in U (and the order of arrivals) are chosen — adversarially, sampled from a known distribution (with a known type graph) in the IID fashion, or via a limited adversary (as in the ROM model where the adversary chooses the graph but not the sequence of arrivals).

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 $u \in M$,

[5] The Goel and Mehta [30] result is even stronger as it holds for the ROM model, but we do not need the stronger result in our paper.

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 a deterministic algorithm ALG in any of the models being considered and a bipartite graph G, 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 For a deterministic algorithm ALG and adversarial input, we define its *asymptotic approximation ratio* as

$$\rho(\mathrm{ALG}) = \lim_{\mathrm{OPT}(G)\to\infty} \inf_G \frac{\mathrm{ALG}(G)}{\mathrm{OPT}(G)}.$$

For a randomized algorithm and distributional input, its *asymptotic approximation ratio* is

$$\rho(\mathrm{ALG}) = \lim_{\mathbb{E}[\mathrm{OPT}(G)]\to\infty} \inf_G \frac{\mathbb{E}[\mathrm{ALG}(G)]}{\mathbb{E}[\mathrm{OPT}(G)]},$$

where the expectation is over the input distribution and the randomness of the algorithm. Note that this includes the special case of the asymptotic approximation ratio of a deterministic algorithm with distributional input.

For a randomized algorithm and adversarial input, its *asymptotic approximation ratio* is

$$\rho(\mathrm{ALG}) = \lim_{\mathrm{OPT}(G)\to\infty} \inf_G \frac{\mathbb{E}[\mathrm{ALG}(G)]}{\mathrm{OPT}(G)},$$

where the expectation is taken over the randomization in the algorithm.

Definition 2 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 by a greedy one. Specifically, if a non-greedy algorithm does not match an online vertex u although u has available neighbors, the greedy simulation will match u to an arbitrary available vertex v. If later the non-greedy algorithm wants to match some online u' to v, then two cases exist. Either there are no other available offline vertices for u', in which case, the size of the matching remains the same and the remaining online and available offline vertices are the same. If instead, u' has another available offline vertex, u' will be arbitrarily matched to some v'. The simulation continues with v' playing the role of v now. Thus, we see that the simulation augments the original matching along an alternating path starting with an online node and an edge not in the original matching of the non-greedy algorithm. Therefore, the matching constructed by the simulation is at least as large as the original matching. See Pena and Borodin [51] for details.

In the bipartite graphs $G = (U, V, E)$ that we consider in this paper, we shall often refer to "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, the graph G as well as an order π of its online vertices U is chosen by an adversary. Karp, Vazirani, and Vazirani [38] presented an optimal randomized algorithm in the adversarial model called Ranking (see Algorithm 1). They showed that $\rho(\textsc{Ranking}) = 1 - 1/e$ and that no randomized algorithm can do better in the adversarial model. Ranking works by choosing a permutation σ of the offline nodes uniformly at random and running the natural simple greedy algorithm on G breaking ties using σ. That is, when an online vertex u arrives, if u has many unmatched neighbors v then u is matched with the v that minimizes $\sigma(v)$.

Let π be a permutation of U in which the nodes are revealed online, and σ be a permutation of V chosen by the algorithm. In this case, we write $\textsc{Ranking}(\pi, \sigma)$ 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 $\textsc{Ranking}(\sigma)$.

Algorithm 1 The RANKING algorithm.

```
procedure RANKING($G = (U, V, E)$)
    Pick a permutation $\sigma : V \to V$ uniformly at random
    for all $u \in U$ do
        When $u$ arrives, let $N(u)$ be the set of unmatched neighbors of $u$
        if $N(u) \neq \emptyset$ then
            Match $u$ with $\arg\min_v \{\sigma(v) \mid v \in N(u)\}$
```

2.2 Known IID Model with Uniform Probability Distribution

In the known IID model, graph $G = (U, V, E)$ is called a *type graph*. Nodes in u are sometimes referred to as "types". The type graph specifies the distribution from which the actual *instance graph* $\widehat{G}$ is generated. An instance graph $\widehat{G} = (\widehat{U}, \widehat{V}, \widehat{E})$ 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 $(u, v) \in E$.

Note that each online $\widehat{u}$ is drawn independently 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. After that, nodes $\widehat{u}$ are generated one by one on the fly and presented to the algorithm one at a time. An online algorithm makes an irrevocable decision on how to match a newly generated $\widehat{u}$ (if at all) without knowledge of the future. We shall sometimes refer to U and $\widehat{U}$ as the online side. Note that the known IID model can be defined more generally by allowing an arbitrary distribution p on U from which $\widehat{u}$ is sampled IID. We don't do that since our (negative) results only need the case when p is the uniform distribution. We make one additional assumption that $|\widehat{U}| = |U|$. This is not necessary, but all our examples are in this setting.

For a given type graph G we shall write μ_G to denote the corresponding distribution defined above. Observe that in this setting, we have $\text{OPT}(G) = \mathbb{E}_{\widehat{G} \sim \mu_G}(M(\widehat{G}))$, and $\text{ALG}(G) = \mathbb{E}_{\widehat{G} \sim \mu_G}(\text{ALG}(\widehat{G}))$.

As already stated, algorithms can be taken to be greedy without loss of generality in the known IID model (see Remark 1) and greedy algorithms obtain approximation at least $1 - \frac{1}{e}$ in the IID model, as long as they satisfy a consistency condition. While the actual definition of the consistency condition is quite technical and appears in Appendix B, it is a very mild restriction. In particular, all algorithms that we consider satisfy the consistency condition. Informally, the condition says that if an online node u is matched with v^* in some run of the algorithm and v^* is available in another "similar" run of the algorithm, u should still be matched with v^*. "Similar" means two things: (1) the new run is obtained by permuting online nodes, and (2) the neighbors of u in the second run form a subset of the neighbors of u in the first run.

Theorem 1 ([30]) *Every consistent greedy algorithm* ALG *in the known IID model satisfies*

$$\rho(\text{ALG}) \geq 1 - 1/e.$$

2.3 Priority Model

In the priority model, an input instance consists of several data items. For the MBM problem, a data item is a pair $(u, N(u))$, where u is the name of an online vertex and $N(u)$ is the set of names of its offline neighbors. Let $\mathcal{I}$ be the infinite universal set of all data items. An input instance is then a finite subset $S = \{(u_i, N(u_i))\} \subseteq \mathcal{I}$ that defines the bipartite graph $G = (U, V, E)$ in a natural way: $U = \{u_i\}, V = \cup_i N(u_i)$, and $(u, v) \in E$ if and only if there exists j such that $u = u_j$ and $v \in N(u_j)$. A *fixed priority* algorithm a priori chooses a permutation $\tau : \mathcal{I} \to \mathcal{I}$ of the entire universe of data items. The adversary then chooses an instance $S \subseteq \mathcal{I}$ and presents it to the algorithm one data item at a time in the order of τ. The algorithm is required to make an irrevocable decision on how to match (if at all) the newly arriving online node u. An *adaptive priority* algorithm can change the permutation τ of the universe of data items after each arriving node u. This gives us two models of deterministic priority algorithms. There are two natural ways of introducing randomness in this model: (1) allow the algorithm to use randomness in selecting τ, (2) allow the algorithm to use randomness for making decisions on how to match the newly arriving node u. This leads to eight different possible priority models. In this paper, we shall be interested in an adaptive priority randomized algorithm called MINRANKING (see Section 5). As defined, MINRANKING is a fully randomized priority algorithm, (i.e. in terms of both the ordering and decisions being randomized) but it can also be modified to be a randomized priority algorithm in the sense that its irrevocable decisions are randomized (as done in the RANKING algorithm) whereas the input sequence is chosen deterministically.

3 Deterministic Multipass Online Algorithms

To break through the deterministic 1/2 barrier in the adversarial setting either the input or the algorithmic model needs to be changed. Dürr et al. [23] consider the model of online algorithms with advice.

We can easily interpret their Category-Advice algorithm as a two-pass algorithm that uses the first pass to create advice (i.e., categories) concerning the offline nodes to be used in the second pass over the input. Their Category-Advice algorithm thus provides a deterministic 2-pass algorithm that achieves a 3/5 approximation ratio. The Category-Advice algorithm 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 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 is performed with σ_c as the permutation of the offline nodes. In other words, a category algorithm partitions the offline nodes into $|\operatorname{Image}(c)|$ categories and specifies the ranking of the categories, the ranking within the category is induced by the initial permutation σ.

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 with 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, as depicted in Figure 7, show that the category function $c(v) =$ degree of v will not yield an asymptotic approximation better than 1/2.)

Our natural extension of Category-Advice to multiple passes is called k-pass Category-Advice and is presented in Algorithm 2. The following theorem is the main result of this section and will follow immediately from Lemmas 4 and 5.

Theorem 2 *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.*

Moreover, the worst-case instances witnessing F_{2k}/F_{2k+1} approximation ratio have the following property: running $k' > k$ passes improves the size of the matching by exactly a single edge. We also observe that to perform an intermediate pass, one needs $O(n \log k)$ total space, i.e., $O(\log k)$ bits per offline vertex to store in which pass a vertex is matched for the first time. Thus, if one is just interested in approximating the size of a matching, the k-pass algorithm can be implemented as a multiple-round semi-streaming algorithm using $O(n)$ bits of space for constant k.

We also show that a natural degree-based category advice algorithm cannot achieve asymptotic approximation ratio greater than 1/2. We finish this section with a formalization of 2-pass category-based "small" advice algorithms and a conjecture that such algorithms (even randomized) cannot achieve approximation ratio greater than $1 - 1/e$.

3.1 Positive Result

The 2-pass algorithm in [23] is called Category-Advice, so we refer to our generalization as k-pass Category-Advice. The pseudocode 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 $-\infty$ 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.

Before we prove the main result of this subsection, we recall several useful results regarding the analysis of the Ranking algorithm.

Algorithm 2 The k-PASS CATEGORY-ADVICE algorithm.

```
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
        Perform the $i$th pass: $M_i \leftarrow$RANKING$(\sigma_c)$
        for $v \in V$ do
            if $c(v) = -\infty$ and $v \in M_i$ then
                $c(v) \leftarrow -i$
    return $M_k$
```

Lemma 1 (Lemma 2 in Birnbaum and Mathieu [8]) *Let $G = (U, V, E)$ be a given bipartite graph. Let π be a permutation of the online side, and σ be a permutation of the offline side. Let $x \in U \cup V$, and set $H = G \setminus x$. Let π_H and σ_H be the permutations of U_H and V_H induced by π and σ. If the matchings* RANKING(π_H, σ_H) *and* RANKING(π, σ) *are not identical, then they differ by a single alternating path starting at x.*

Let M be a maximum matching in G. Choose $x \in V \setminus V(M)$ and delete it. The above lemma implies that the size of the matching found by RANKING on H is at most the size of the matching found by RANKING on G, while the size of a maximum matching hasn't changed. We can repeat this procedure until the resulting graph has a perfect matching M. This means that the worst-case approximation ratio of RANKING is achieved on bipartite graphs with a perfect matching. Since the k-PASS CATEGORY-ADVICE algorithm is a RANKING-based algorithm, the same conclusion holds for k-PASS CATEGORY-ADVICE. Hence, from now on we assume that the input graph $G = (U, V, E)$ is such that $|U| = |V| = n$ and G has a perfect matching. Before proving the main result of this subsection, we need one more lemma regarding RANKING.

Lemma 2 (Implicit in the proof of Lemma 4 in Birnbaum and Mathieu [8]) *Let σ_1 be a permutation of the offline vertices and let $M_1 =$*RANKING(σ_1). *Let v be an offline vertex such that $v \notin M_1$. Let σ_2 be a permutation of the offline vertices obtained from σ_1 by improving the rank of v. Let $M_2 =$*RANKING(σ_2). *Then for every online node u, if $u \in M_1$ then $u \in M_2$ and $\sigma_2(M_2(u)) \leq \sigma_2(M_1(u))$.*

This lemma immediately implies the following:

Lemma 3 *Let M_k be the matching obtained by running the k-*PASS CATEGORY-ADVICE *algorithm on some instance. Let $M_{k'}$ be the matching obtained by running the k'-*PASS CATEGORY-ADVICE *algorithm on the same instance. If $k \leq k'$ then $|M_k| \leq |M_{k'}|$.*

Proof Simple induction.

We are now ready to prove the main result of this subsection.

Lemma 4 *The k-*PASS CATEGORY-ADVICE *algorithm achieves approximation ratio F_{2k}/F_{2k+1}.*

Proof (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. Assume that the k-PASS CATEGORY-ADVICE algorithm achieves approximation ratio F_{2k}/F_{2k+1}. Let's consider the $(k + 1)$-PASS CATEGORY-ADVICE algorithm running on some bipartite input graph $G = (U \cup V, E)$, where U are the online nodes. By Lemma 1, we may assume without loss of generality that $|U| = |V|$ and G has a perfect matching. 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 computing a bound on $|M_{k+1}| = |M_{11}| + |M_{12}| + |M_{21}|$. Figure 1 is a graphical depiction of the variables described.

Observe that 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 exactly the same as if we ran k-PASS CATEGORY-ADVICE algorithm on the subgraph of G induced by $U_1 \cup V_2$. By induction this implies that $|M_{12}| \geq (F_{2k}/F_{2k+1})|V_2| = (F_{2k}/F_{2k+1})|U_2|$.

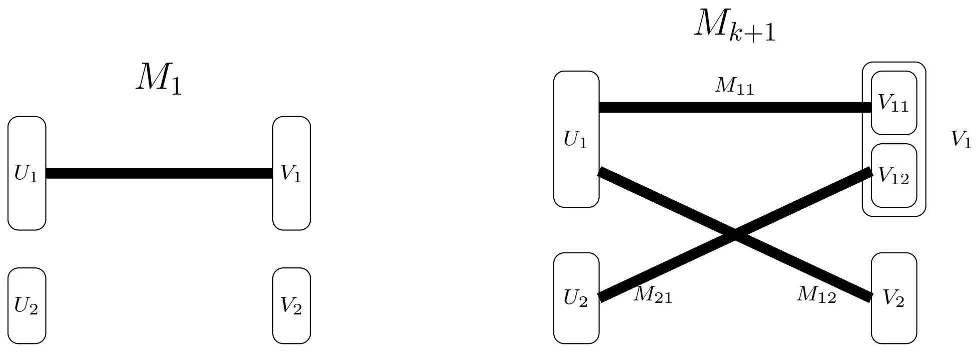

Fig. 1 G is the input graph. On the left we show the matching constructed in the first pass. On the right we show the matching constructed in the $k+1$st pass.

By Lemma 2, 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 $V_{12} = V_1 \setminus V_{11}$. We would like to lower bound $|M_{21}|$. To that end 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 we assume 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, observe that 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

$$\begin{aligned}|M_{k+1}| &= |M_{11}| + |M_{12}| + |M_{21}| = |U_1| - |M_{12}| + |M_{12}| + |M_{21}| \\ &\geq |U_1| + \frac{1}{2}(|U_2| - |U_1| + |M_{12}|) \geq \frac{1}{2}\left(|U_2| + |U_1| + \frac{F_{2k}}{F_{2k+1}}|U_2|\right) \\ &= \frac{1}{2}\left(n + \frac{F_{2k}}{F_{2k+1}}(n - |M_1|)\right) = \left(\frac{1}{2} + \frac{F_{2k}}{2F_{2k+1}}\right)n - \frac{F_{2k}}{2F_{2k+1}}|M_1|.\end{aligned}$$

By Lemma 3 we also have that $|M_{k+1}| \geq |M_1|$. Thus, we derive

$$|M_{k+1}| \geq \max\left\{|M_1|, \left(\frac{1}{2} + \frac{F_{2k}}{2F_{2k+1}}\right)n - \frac{F_{2k}}{2F_{2k+1}}|M_1|\right\}.$$

This implies that $|M_{k+1}| \geq \frac{1/2 + F_{2k}/2F_{2k+1}}{1 + F_{2k}/(2F_{2k+1})}n = \frac{F_{2k+1} + F_{2k}}{2F_{2k+1} + F_{2k}}n = \frac{F_{2(k+1)}}{F_{2(k+1)+1}}n$.

3.2 Negative Result

In this subsection, we construct a family of bipartite graphs $(G_k)_{k=1}^{\infty}$ with the following properties:

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 $F_{2k} + 1$.

Remark 2 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.

This family of graphs shows that our analysis of the k-pass Category-Advice algorithm is tight. First, we describe the construction and then prove the above properties.

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.

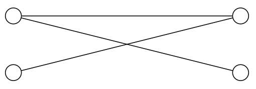

Fig. 2 G_1 is the graph used for the basis of the induction.

In the recursive step, we assume that G_k has been constructed, and we show how to construct G_{k+1}. 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.

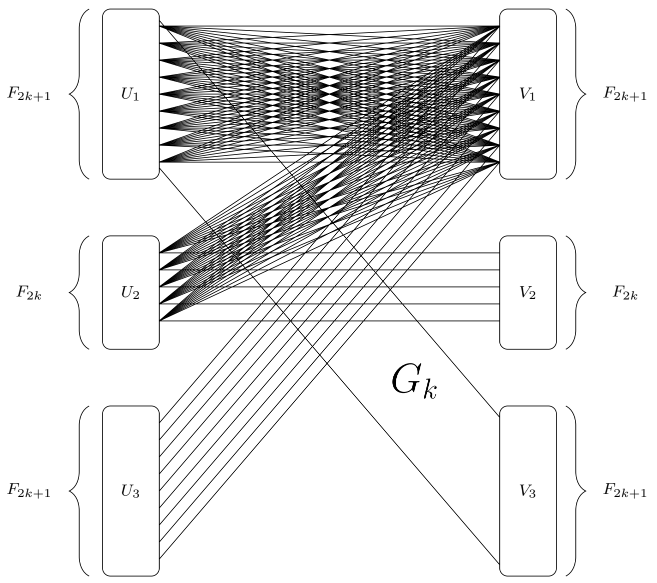

Fig. 3 G_{k+1} depicts the inductive construction.

Lemma 5 *Properties (1-4) mentioned at the beginning of this subsection hold for the G_k.*

Proof (Proof by induction on k.)
Base case: $k = 1$. We have that G_1 has 2 online and 2 offline nodes and $F_3 = 2$. It is easy to see that G_1 has a perfect matching. The 1-PASS CATEGORY-ADVICE algorithm is simply the regular deterministic greedy algorithm, which clearly returns a matching of size 1 on G_1, and $F_2 = 1$. Completing the base case, 2-PASS CATEGORY-ADVICE finds a matching of size $F_2 + 1 = 2$ and adding more passes after that does not change the matching found by the algorithm.

Inductive step. Assume that properties (1-4) hold for G_k. First, observe that 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. Recall, that the order in which online vertices U appear is the top-to-bottom order in Figure 3. 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 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). Also note that 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 similar to the third property. Let $k' > k$ be given and we analyze the behavior of the k'-PASS CATEGORY-ADVICE on G_{k+1}. After the first pass, by inductive assumption, the next $k'-1$ passes are going to produce a matching of size $F_{2k}+1$ between U_1 and V_3. This means that in the last pass, $F_{2k+1} - (F_{2k}+1)$ vertices from U_1 will be matched with V_1. This, in turn, implies that F_{2k} vertices from U_2 will be matched with V_1, leaving 1 more vertex from V_1 to be matched with U_3. This results in the overall matching of size $F_{2k+1} + F_{2k} + 1 = F_{2k+2} + 1$.

3.3 Concluding Remarks Regarding Category-Based Algorithms

Since our later results in the paper explore the power of degree-based tie-breaking in bipartite matching, we mention a natural 2-pass category algorithm based on the degrees of the offline vertices. Namely, in the first pass this algorithm computes the degrees of offline nodes and assigns those degrees as categories. In the second pass the algorithm runs RANKING on the permutation induced by the categories in the same manner as in the 2-PASS CATEGORY-ADVICE algorithm. A simple modification of a typical worst-case instance for online algorithms shows that the degree-based category algorithm does not achieve asymptotic approximation ratio greater than $1/2$ — see Figure 4.

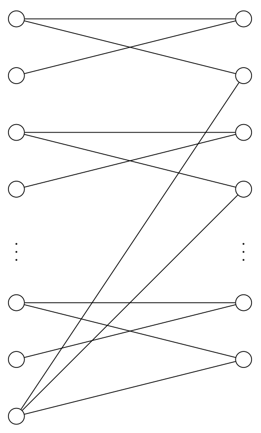

Fig. 4 A simple modification of a worst-case instance for the online algorithm that witnesses a poor performance of the degree-based category algorithm. Offline nodes are on the right and all have degree 2.

Next, we describe a formalization of an ℓ-bit category-based algorithm. Based on the entire input graph, an algorithm assigns ℓ-bit labels to the offline nodes. Then, the algorithm runs RANKING with the permutation induced by the categories (the same way as the 2-PASS CATEGORY-ADVICE algorithm). Observe that our k-PASS CATEGORY-ADVICE algorithm is an instance of a $\lceil \log k \rceil$-bit category-based algorithm. We conjecture[6] that for constant ℓ even if an ℓ-bit category-based algorithm is allowed to make randomized decisions in assigning categories, it cannot achieve an approximation ratio better than $1 - 1/e$. Such a sequence of randomized algorithms with approximation ratio converging to $1 - 1/e$ (as ℓ goes to infinity) was presented in [23].

4 Analysis of MinDegree in the Known IID Model

In this section we prove a tight approximation ratio for a natural greedy algorithm in the known IID input model—MINDEGREE. Recall from Section 2 that every algorithm achieving approximation ratio ρ in the known IID model can be converted into a greedy algorithm achieving approximation ratio

[6] We note that an ℓ-bit category advice algorithm is a restricted form of an ℓn bit advice algorithm and hence this conjecture is incomparable with the $\log \log n$ lower bound on advice bits for deterministic online algorithms that is proven in [51].

$\geq \rho \geq 1 - \frac{1}{e}$. Ties may occur in a greedy algorithm when an online node has more than one available neighbor. A greedy algorithm is precisely defined when we specify a tie-breaking rule. How important is the tie-breaking for the performance of a greedy algorithm in the known IID model? Turns out that it's very important and a good tie breaking rule can improve the approximation ratio.

Algorithms beating $1-1/e$ ratio are known in this model ([5,46,31,35,15]). Although these algorithms are not stated as greedy algorithms, we can apply the general conversion to turn them into greedy algorithms, albeit with unnatural tie-breaking conditions. The tie-breaking rules of these algorithms often require polynomial time preprocessing of the type graph, thus they are feasible from the theoretical point of view, but not from a practical point of view for large type graphs. This motivates the study of conceptually simple and practical tie-breaking rules for greedy algorithms in the known IID model. We contribute to this by studying a natural tie-breaking rule.

Remark 3 The following is folklore. In the known IID model, there is a dynamic program (DP) computing a tie-breaking rule that gives an online algorithm having the best approximation ratio among all online algorithms. We shall refer to this tie-breaking rule as the *optimal tie-breaking rule.* The entry (V, n) of the DP table T stores the expected size of a matching achieved by a best online algorithm when the next n online nodes are generated from the type graph $G = (U, V, E)$. Observe that an online node u is matched with its neighbor $v \in V$ that maximizes the value of the DP table at $(V \setminus \{v\}, n-1)$. Thus, the recursive definition of T is

$$T[V, n] = \mathbb{E}_u \left(\max \left(T[V, n-1] \ , \ \mathbb{I}(N(u) \neq \emptyset) + \max_{v \in N(u)} T[V \setminus \{v\}, n-1] \right) \right).$$

$$T[\emptyset, n] = T[V, 0] = 0$$

Here $\mathbb{I}(S)$ is the indicator function which is equal to 1 if statement S is true and 0 otherwise. Unfortunately, the size of this dynamic programming table is exponentially large, so computing this optimal tie-breaking rule using such a dynamic program is not computationally feasible. Currently no one knows how to analyze the optimal tie-breaking rule, but certain potentially sub-optimal and computationally feasible tie-breaking rules have been analyzed as discussed above.

Our algorithm is MinDegree. 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 (i.e. make worse) the approximation ratio of the algorithm. Generalizing this, we may come to the conclusion that online vertices should give preference to the offline vertices of smaller degrees. The problem is that in the adversarial setting we do not know the degrees of the offline nodes 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. The pseudocode of MinDegree is presented in Algorithm 3.

Algorithm 3 The MinDegree algorithm.

```
procedure MinDegree($G = (U, V, E)$)
    Let $S = V$ denote the set of active offline nodes
    repeat
        Let $\widehat{u}$ denote the newly 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 4 Algorithm 3 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 analysis, we will initially assume that the adversary can decide how the algorithm breaks these ties. We will then show how the argument can be modified to accommodate any tie-breaking by the algorithm.

MinDegree is a conceptually simple and promising algorithm in the known IID setting. Indeed, a version of MinDegree has been extensively studied in the offline setting (see Besser and Poloczek [7] and references therein). Unfortunately, in spite of having excellent empirical performance as well as excellent performance under various random input models, it has a bad worst-case approximation ratio of 1/2 in the offline adversarial setting [7]. As far as we know, MinDegree has not been analyzed in the known IID model. We obtain a result that, in spirit, is similar to the offline worst-case 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.

Theorem 3

$$\rho(\textsc{MinDegree}) = 1 - \frac{1}{e}$$

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

First observe that $\rho(\textsc{MinDegree}) \geq 1 - 1/e$ immediately follows from Theorem 1, since it is easy to see that MinDegree satisfies the consistency condition in Definition 4. Thus, the rest of this section is dedicated to proving $\rho(\textsc{MinDegree}) \leq 1 - 1/e$. Namely, we prove the following.

Lemma 6 (MinDegree negative result) *There is a family of type graphs witnessing*

$$\rho(\textsc{MinDegree}) \leq 1 - 1/e.$$

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

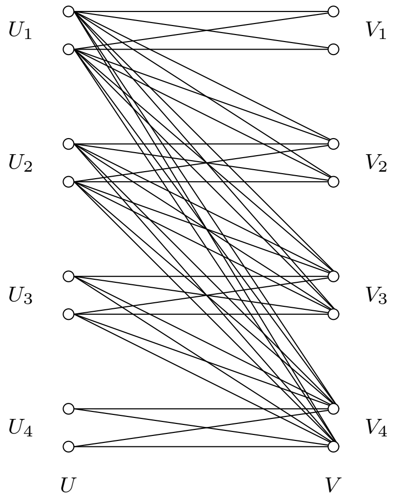

Fig. 5 An example of $G_{L,N}$ with $L = 2, N = 4$. Such graphs are used as building blocks for the family of hard type graphs for MinDegree.

To prove the above theorem we shall construct a type graph with the online side of size $LNK + LN$ and the offline side of size $LNK + o(LNK)$. The actual number of online nodes to be generated is LNK. The parameters L, N, and K are explained below. We shall often use $o(f(n))$ notation to simplify the presentation. Not every choice of functions in $o(f(n))$ makes the argument work, but it should be clear from the proof that *there is* a choice of $o(f(n))$ functions that work. This existential argument is sufficient for our purposes.

First, we use the family of type graphs due to Goel and Mehta [30] as a building block. Each graph in that family is indexed by two integers L and N. Let $G_{L,N}(U, V, E)$ denote the (L, N)th graph in the family. The offline side is partitioned into N blocks of size L: $V = \dot{\bigcup}_{i=1}^{N} V_i$. The online side is partitioned into N blocks of size L, i.e., $U = \dot{\bigcup}_{i=1}^{N} U_i$. The edges are as follows: there is a biclique between V_i and U_j if and only if $i \geq j$. An example of this graph with $L = 2$ and $N = 4$ is depicted in Figure 5.

Theorem 4 (Goel and Mehta [30]) *There is a tie-breaking criterion such that a simple greedy algorithm finds a matching of size $\leq LN(1 - 1/e) + o(LN)$ on $G_{L,N}$ with high probability.*

In fact, the implicit tie-breaking criterion in the work of Goel and Mehta on $G_{L,N}$ is equivalent to breaking ties by maximum degree, i.e., online nodes from U_j are matched with a neighboring node from V_i with largest possible i. Thus, the online nodes try to fill in the spaces on the V side from bottom up to prevent future online nodes from having neighbors. The high level idea behind our construction is to use K independent copies of $G_{L,N}$ and N shared gadgets. The gadgets are used to make the offline degrees the same on each copy of $G_{L,N}$ without affecting the size of the matching found by the algorithm on a particular copy of $G_{L,N}$. Then we can invoke an adversarial argument to say that MinDegree on the augmented $G_{L,N}$ behaves exactly like the Goel and Mehta greedy algorithm on the original $G_{L,N}$. As the number of copies K goes to infinity, the contribution of the shared gadgets to the size of the overall matching goes to zero. Thus, we get the approximation ratio of $1 - 1/e$ for MinDegree.

We denote our type graph by $G_{L,N,K}$. Let $G^i_{L,N}$ denote the ith copy of $G_{L,N}$ in our type graph for $i \in [K]$. Gadget j, denoted by $H_{j,L}$, simply consists of a biclique with the offline side of size $L + o(L)$ and the online side of size L. We let H_L denote $\dot\bigcup_{j=1}^{N} H_{j,L}$. For each i, we connect the offline side of the kth block of $G^i_{L,N}$ via a biclique with the online side of $H_{j,L}$ for $k \leq j$. Figure 6 shows an example, where the offline side has the same number of nodes as the online side in each gadget for simplicity. In Figure 6, black nodes indicate the online side and white nodes indicate the offline side.

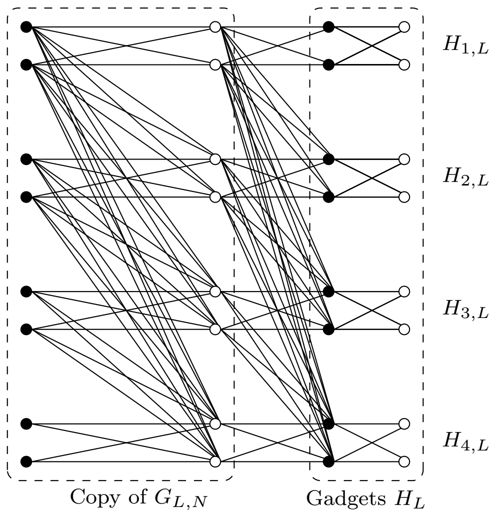

Fig. 6 An example of how gadgets $H_{j,L}$ connect to a copy of $G_{L,N}$ in the family of hard type graphs. Here, we have $L = 2, N = 4$, black nodes indicate the online side, and white nodes indicate the offline side.

Proof (Proof of Theorem 6)
Since each offline node in each copy $G^i_{L,N}$ has the same degree, we will initially assume that MinDegree breaks ties by matching an online node from block j with an offline node of block k for the largest possible k. This is the same behavior as in Goel and Mehta [30] analysis. Assume that L, N, and K are large enough. By the Chernoff bound, with high probability the number of online nodes generated from $H_{j,L}$ is bounded by $L+o(L)$. Thus, by the choice of our gadget, with high probability all nodes from $H_{j,L}$ can be matched to the offline nodes within $H_{j,L}$. Therefore, the online nodes generated from gadgets do not have affect on any of the independent copies of $G_{L,N}$. Moreover, the expected number of nodes generated from $G^i_{L,N}$ is LN. Invoking the Chernoff bound again, we see that with high probability for each i we generate at most $LN + o(LN)$ nodes from $G^i_{L,N}$. Since independent copies $G^i_{L,N}$ do not share any edges except with the gadget, the matchings constructed by our algorithm on independent copies are disjoint. Since the behavior of our algorithm on each copy of $G_{L,N}$ is identical to the behavior of a simple greedy algorithm, our algorithm for each $G^i_{L,N}$ constructs a matching of expected size $LN(1 - 1/e) + o(LN)$. Thus, overall our algorithm finds a matching of expected size $LNK(1 - 1/e) + o(LNK)$. Moreover, it is easy to see that a matching of size at least $LNK - o(LNK)$ is possible. The claimed approximation ratio follows.

Remark 5 The claim of Theorem 6 can be strengthened by taking away the power of the adversary in breaking ties. We can force the unfavorable tie-breaking used in the above proof by introducing an additional biclique gadget with $N + o(N)$ offline nodes and N online nodes. We modify the degrees in

$G^i_{L,N}$ in a natural way: the offline nodes from block j of $G^i_{L,N}$ get connected to $N-j$ nodes of the online side of the new gadget. This forces the MinDegree algorithm to process blocks $G^i_{L,N}$ from "bottom to top" (see Figure 5), which is exactly the wrong order that results in $LNK(1-1/e)+o(LNK)$ matching. It is easy to see that the only other affect this additional gadget introduces is to increase the size of the matching found by MinDegree, as well as OPT, by a lower order term $N+o(N)=o(LNK)$.

5 A Hybrid Algorithm in the Priority Model

We propose a conceptually simple greedy algorithm for bipartite matching. Our algorithm is a natural hybrid between two well-studied greedy algorithms for bipartite matching—Ranking (see Section 2) and MinGreedy (due to Tinhofer [56]). 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. Despite excellent empirical performance and excellent performance under certain random input models[7], the algorithm achieves approximation ratio 1/2 in the adversarial input setting (see [7] and references therein).

In the bipartite setting, we shall use MinGreedy to refer to an algorithm that 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.

Karp, Vazirani, Vazirani [38] exhibited a family of graphs, on which Ranking achieves its worst approximation ratio $1-1/e$. The biadjacency matrix of the nth graph in this family is an $n\times n$ upper triangular matrix with all 1s at and above the diagonal. Interestingly, 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 chosen. Then, nodes in U are processed in increasing order of their current degrees with ties broken randomly. When a node u is processed, it is matched with its neighbor appearing earliest in the ordering π.

Algorithm 4 The MinRanking algorithm.

```
procedure MinRanking($G = (U, V, E)$)
    Pick a permutation $\pi : V \to V$ uniformly at random
    repeat
        Let $d = \min\{\text{cur deg}(i) \mid i \in U\}$ and $S = \{i \in U \mid \text{cur deg}(i) = d\}$
        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
            Match $i$ with $j = \arg\min_k \{\pi(k) \mid k \in N(i)\}$
            Delete $i$ and $j$ from $G$
            Update cur deg
    until $U = \emptyset$
```

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 in [38] 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 [13]. Since our algorithm is a generalization of Ranking, its asymptotic approximation ratio is at least $1-1/e\approx 0.6321$. The main result of this section is that the asymptotic approximation ratio of this algorithm is at most $1/2+1/(2e)\approx 0.6839$, as witnessed by the family of graphs due to Besser and Poloczek [7].

[7] The performance of MinGreedy and related degree based algorithms is mainly done in the offline model for general graphs and then with respect to synthetically generated data. We are not aware of publicly available benchmark data for bipartite matching instances in an online model. Yet the intuition remains that choosing vertices according to minimum degree is a good heuristic.

Theorem 5

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

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 [45]. From our result it follows that a deterministic ordering of the online nodes by non-decreasing degree (breaking ties by the given adversarial ordering of those nodes) will also fall short. That is (similar to the result in [51] for deterministic decisions), a naive randomized ordering can be better than a seemingly informed deterministic ordering.

5.1 The Besser-Poloczek Graph Construction

In this section we describe the family of bipartite graphs constructed by Besser and Poloczek in [7]. Each graph in the family is indexed by an integer parameter b. Let $G_b(L,R,E)$ denote the bth graph in the family. A note about notation: we use L and R instead of U and V in the definition of G_b. This is because we want to reserve U and V for a different graph that will play the central role in the analysis of MinRanking. Continuing with the definition of $G_b(L,R,E)$, each of the two sides L and R is partitioned into three sets:

- $L = S_{1,L} \,\dot\cup\, S_{2,L} \,\dot\cup\, S_{3,L}$
- $R = S_{1,R} \,\dot\cup\, S_{2,R} \,\dot\cup\, S_{3,R}$

Blocks $S_{2,L}$ and $S_{2,R}$ are further partitioned into b sets each:

- $S_{2,L} = \dot\bigcup_{i=1}^{b} S_{2,L}^{(i)}$
- $S_{2,R} = \dot\bigcup_{i=1}^{b} S_{2,R}^{(i)}$

For convenience, we give numeric names to each vertex in L and R:

- $S_{1,W} = \{1_W, 2_W, \ldots, b^2_W\}$ where $W \in \{L,R\}$
- $S_{2,W}^{(i)} = \{(b^2+(i-1)b+1)_W, \ldots, (b^2+ib)_W\}$ where $W \in \{L,R\}$
- $S_{3,W} = \{(2b^2+1)_W, \ldots, (2b^2+2b)_W\}$ where $W \in \{L,R\}$

Thus, we have $|L| = |R| = 2b^2+2b$. Next, we describe the edges of G_b:

- There is a biclique between $S_{3,L}$ and $S_{1,R}$ and between $S_{3,R}$ and $S_{1,L}$.
- $S_{1,R}$ and $S_{2,L}$ are connected by "parallel" edges, i.e., $\{i_R, (b^2+i)_L\} \in E$ for all $i \in [b^2]$.
- $S_{1,L}$ and $S_{2,R}$ are connected by "parallel" edges, i.e., $\{i_L, (b^2+i)_R\} \in E$ for all $i \in [b^2]$.
- $S_{3,L}$ and $S_{3,R}$ are connected by "parallel" edges, i.e., $\{i_R, i_L\} \in E$ for all $i \in \{2b^2+1, \ldots, 2b^2+2b\}$.
- There is a biclique between $S_{2,L}^{(i)}$ and $S_{2,R}^{(i)}$ for each $i \in [b]$.

Figure 7 depicts the Besser-Poloczek graph with $b=3$. We note that G_b has a perfect matching.

Claim G_b has a perfect matching, i.e., a matching of size $2b^2+2b$.

Proof Using the parallel edges, match the vertices in $S_{1,R}$ with the vertices in $S_{2,L}$, the vertices in $S_{2,R}$ with the vertices in $S_{1,L}$, and the vertices in $S_{3,L}$ with the vertices in $S_{3,R}$.

Besser and Poloczek [7] show that the graphs G_b are asymptotically hardest for MinGreedy.

Theorem 6 (Besser,Poloczek [7]) *The expected size of a matching constructed by* MinGreedy *on* G_b *is* $b^2+o(b^2)$. *Thus, the worst case asymptotic approximation ratio of* MinGreedy *is* $1/2$.

In the next section we shall see that MinRanking finds a significantly larger matching on G_b.

5.2 Analysis of MinRanking on the Besser-Poloczek Graphs

This section is devoted to proving the following theorem, which immediately implies Theorem 5.

Theorem 7 MinRanking *running on the family of graphs* $\{G_b\}$ *has the asymptotic approximation ratio exactly* $\frac{1}{2}+\frac{1}{2e}$.

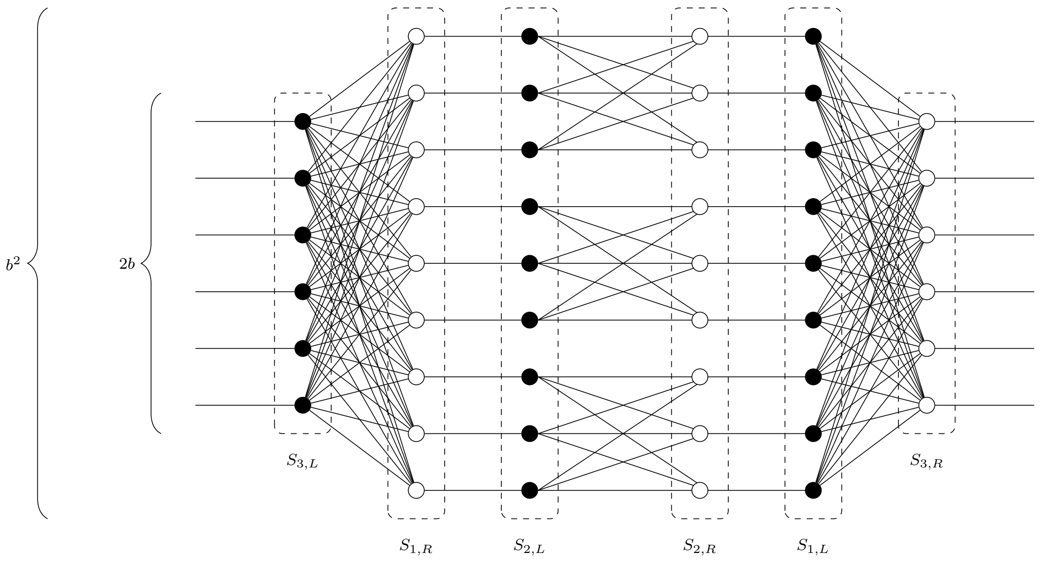

Fig. 7 The Besser-Poloczek graph with parameter $b = 3$. The edges on the left wrap around on the right connecting the nodes in $S_{3,L}$ to the nodes in $S_{3,R}$.

We shall prove the above theorem in two steps. In the first step, we get tight bounds on the expected number of edges of relevant type in a matching constructed by MinRanking running on certain induced subgraphs of the Besser-Poloczek graphs. In the second step, we show how to combine the results for subgraphs and derive the expected size of a matching of the entire Besser-Poloczek graph.

We begin by analyzing the expected performance of MinRanking on a particular bipartite graph $H_{n,k}(U, V, W)$ parametrized by two integers n and k where $k \leq n$. The relevance of this analysis to Theorem 7 will be evident later. We use a different capital letter for these graphs to distinguish them from the Besser-Poloczek graphs. The left side consists of n vertices $U = \{1_U, \ldots, n_U\}$. The right side is partitioned into two blocks $V = V_1 \dot{\cup} V_2$ such that the first block contains k vertices $V_1 = \{1_{V_1}, \ldots, k_{V_1}\}$ and the second block contains n vertices $V_2 = \{1_{V_2}, \ldots, n_{V_2}\}$. The edges consist of biclique edges between U and V_1 and parallel edges between U and V_2, i.e., we have $\{i_U, i_{V_2}\} \in W$ for $i \in [n]$. Figure 8 depicts this bipartite graph for $n = 5$ and $k = 3$.

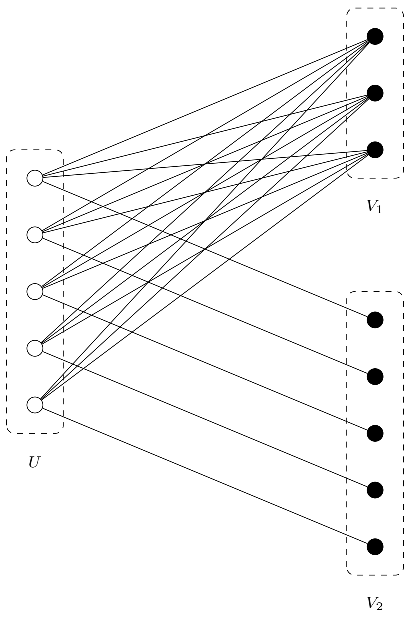

Fig. 8 The $H_{n,k}$ graph with $n = 5$ and $k = 3$.

Consider MinRanking running on $H_{n,k}$. For reasons that will become clear later, we are interested in obtaining sharp bounds on the expected number of (U, V_2) edges in a matching output by MinRanking. Let $t \in \{0, 1, \dots, n\}$ denote the time, i.e., the iteration number of the main loop in MinRanking with 0 being the time prior to the execution of the main loop. First, we make the following simple observation.

Lemma 7 *Consider* MinRanking *running on $H_{n,k}$. Then $\forall t \in \{0, 1, \dots, n\}$ the subgraph remaining at time t in* MinRanking *(disregarding isolated vertices) is isomorphic to $H_{n-t,k-k'}$ for some $k' \leq k$. In particular, $\forall t \in \{0, 1, \dots, n\}$ every unmatched vertex in U has the same current degree at time t.*

Proof This follows by a simple induction on t. Isolated vertices do not affect the performance of MinRanking, so we freely disregard them whenever they occur. Case $t = 0$ is trivial. By the induction hypothesis, at time t we have a graph isomorphic to $H_{n-t,k-k'}$. In the inductive step, there are possibly two cases at time $t + 1$. In the first case, a vertex in U is matched with a vertex in V_2. After deleting both of these vertices, the remaining subgraph is isomorphic to $H_{n-t-1,k-k'} = H_{n-(t+1),k-k'}$. In the second case, a single vertex u in U is matched with a vertex in V_1 (if there is one remaining, i.e., if $k - k' \geq 1$). After deleting both of these vertices, the unique vertex in V_2 that used to be the neighbor of u becomes isolated. Disregarding this isolated vertex, the remaining subgraph is isomorphic to $H_{n-t-1,k-k'-1} = H_{n-(t+1),k-(k'+1)}$.

Therefore, by symmetry, we may assume from now on that MinRanking processes the vertices in U in the order from 1_U to n_U. This assumption does not affect the expected number of (U, V_2) edges in a matching output by MinRanking.

Definition 3 Let RHSGreedy be the algorithm that receives vertices from $V_1 \cup V_2$ in a random order and matches the received vertex to the first available vertex in U, if there is one.

It is easier to analyze the expected number of (U, V_2) edges for RHSGreedy than for MinRanking. Fortunately, the two algorithms produce the same output. The following is an adaption of the "duality principle" as discussed in [38].

Lemma 8 *For a given fixed permutation π of vertices in $V_1 \cup V_2$ the matching constructed by* MinRanking *is exactly the same as the matching constructed by* RHSGreedy *on $H_{n,k}$. Consequently, the expected number of (U, V_2) edges in the output is the same for the two algorithms.*

Proof Let $v_1, \dots, v_{n+k}$ be the vertices in $V_1 \cup V_2$ listed in the order given by π. By Lemma 7, we may assume that the order in which MinRanking considers vertices in U is given by $1_U, \dots, n_U$. Note that RHSGreedy has $n+k$ iterations, since in iteration i it considers v_i, while MinRanking has n iterations, since in iteration i it considers i_U. Thus, the two algorithms run in different time domains. However, if we ignore iterations in which RHSGreedy considers isolated vertices, it is easy to see that both algorithms have n iterations. We prove that the matching constructed by MinRanking is the same as the matching constructed by RHSGreedy by induction on the number of U vertices matched by MinRanking.

Let v_j denote the vertex matched by MinRanking with 1_U. The vertex 1_U can be matched either with a vertex in V_1 or with 1_{V_2}. We consider these two cases separately.

Suppose that $v_j \in V_1$. By the definition of MinRanking, there exists $k > j$ such that $1_{V_2} = v_k$, and for all $i < j$ we have $v_i \in V_2$. Therefore, RHSGreedy on this input matches each $v_i \in V_2$ with its unique neighbor in U for $i < j$. Then v_j will be matched with 1_U, as it will be the first available neighbor of v_j in U. Note that removing the two matched vertices together with 1_{V_2} (which becomes isolated) does not affect the matchings constructed by the two algorithms on the resulting subgraphs. Thus, we can eliminate these vertices and proceed inductively.

Now, suppose that $v_j \in V_2$. By the definition of MinRanking, for all $i < j$ we have $v_i \in V_2$. Arguing as above, RHSGreedy on this input matches each $v_i \in V_2$ with its unique neighbor in U for $i \leq j$. In particular, v_j is matched with 1_U. Eliminating the two matched vertices does not affect the rest of the matching constructed by the two algorithms. Thus, we can proceed inductively.

We showed that in both cases both algorithms include the same edge $(1_U, v_j)$ in their outputs, and we can proceed inductively after removing $1_U, v_j$ and possibly 1_{V_2}.

Lemma 9 *Consider* RHSGreedy *running on $H_{n,n}$. Let $X_n(t)$ denote the number of vertices in U unmatched at iteration t (ignoring trivial iterations with isolated vertices). Let $Y_n(t)$ denote the number of vertices in V_2 that have been matched up to and including iteration t (ignoring trivial iterations with* *isolated vertices). Then $(X_n(t), Y_n(t))$ is a Markov chain with the state space $\{0, 1, \ldots, n\}^2$ and the following transition probabilities:*

$$P((X_n(t+1), Y_n(t+1)) = (x-1, y) \mid (X_n(t), Y_n(t)) = (x, y)) = \frac{x+y}{2x+y}, \text{ and}$$

$$P((X_n(t+1), Y_n(t+1)) = (x-1, y+1) \mid (X_n(t), Y_n(t)) = (x, y)) = \frac{x}{2x+y},$$

where $t \in \{0, 1, \ldots, n-1\}$. The initial state is $(X_n(0), Y_n(0)) = (n, 0)$.

Proof Let $X_n(t) = x$ and $Y_n(t) = y$. Then the number of vertices from V_2 that remain as possible candidates for a match with a vertex in U is x. The number of vertices that have been matched in V_1 is $n - x - y$. Thus, the number of vertices in V_1 that remain as possible candidates for a match with a vertex in U is $n - (n - x - y) = x + y$. In the next non-trivial iteration, the number of remaining U vertices goes down by one, hence $X_n(t+1) = x - 1$. The number of matched vertices in V_2 can either remain the same or increase by one. RHSGreedy matches a vertex from V_2 with a vertex in U if and only if a vertex from V_2 appears next in the permutation of the remaining vertices from $V_1 \cup V_2$, which happens with probability $x/(2x+y)$. Therefore, we have

$$P((X_n(t+1), Y_n(t+1)) = (x-1, y+1) \mid (X_n(t), Y_n(t)) = (x, y)) = \frac{x}{2x+y}.$$

The probability of the other case, i.e., $Y_n(t+1) = y+1$), is simply $1 - x/(2x+y) = (x+y)/(2x+y)$.

By Lemma 8, the expected number of (U, V_2) edges in a matching constructed by MinRanking is equal to $\mathbb{E}(Y_n(n))$. The asymptotic behavior of $\mathbb{E}(Y_n(n))$ is captured by a solution to a related ordinary differential equation – this is an application of the differential equation method (see [40], [57], [58]). We get the following result.

Lemma 10 *Let the setting be as in Lemma 9. For every $\epsilon > 0$ we have*

$$(1-\epsilon)\frac{n}{e} + o(n) \le \mathbb{E}(Y_n(n)) \le \frac{n}{e} + o(n).$$

We remark that $\mathbb{E}(Y_n(n))$ is also equal to the expected number of (U, V_2) edges in a matching constructed by MinRanking.

Proof Define $\Delta Y_n(t) = Y_n(t) - Y_n(t-1)$ and $\Delta X_n(t) = X_n(t) - X_n(t-1)$. Then by Lemma 9 we have

$$\mathbb{E}(\Delta Y_n(t)) = \frac{X_n(t)}{2X_n(t) + Y_n(t)}, \text{ and}$$

$$\mathbb{E}(\Delta X_n(t)) = -1.$$

Thus, we have

$$\frac{\mathbb{E}(\Delta Y_n(t))}{\mathbb{E}(\Delta X_n(t))} = \frac{-X_n(t)}{2X_n(t) + Y_n(t)}.$$

Applying the differential equation method (see [40], [57], [58]), the asymptotics of $Y_n(n)$ are captured by a solution to the following ordinary differential equation (y is regarded as a differentiable function of x):

$$\frac{dy}{dx} = \frac{-x}{2x+y}$$

with the initial condition $y(n) = 0$. Let $\tilde{y}(x)$ be defined implicitly as a solution to the following equation:

$$\ln(1 + \tilde{y}(x)/x) - \frac{1}{1 + \tilde{y}(x)/x} = -1 + \ln(n) - \ln(x).$$

Then it is straightforward to verify that $\tilde{y}(x)$ is a solution to the above ODE. Note that the function $\tilde{y}(x)$ is undefined at $x = 0$ (this essentially comes from the condition of Lemma 9 that $t \le n-1$). However, we have $|\mathbb{E}(Y_n(n)) - \mathbb{E}(Y_n(n-1))| \le 1$, thus we can estimate the asymptotics of $\mathbb{E}(Y_n(n))$ by the asymptotics of $\mathbb{E}(Y_n(n-1))$, which amounts to finding the value of $\tilde{y}$ at $x = 1$:

$$\ln(1 + \tilde{y}(1)) - \frac{1}{1 + \tilde{y}(1)} = -1 + \ln(n).$$

Define

$$f(z) = \ln(1+z) - \frac{1}{1+z} + 1 - \ln(n).$$

We have

$$\begin{aligned} f(n/e) &= \ln(1+n/e) - \ln(n/e) - \frac{1}{1+n/e} \\ &= \ln(1+e/n) - \frac{1}{1+n/e} \\ &= \left(\frac{e}{n} - \frac{e^2}{2n^2} + o(n^{-3})\right) - \left(\frac{e}{n} - \frac{e^2}{n^2} + o(n^{-3})\right) \\ &= \frac{e^2}{2n^2} + o(n^{-3}) \\ &> 0 \end{aligned}$$

where the third equality follows from the Taylor series for $\ln(1+x)$ and $1/(1+1/x)$ around 0, and the last inequality holds for large enough n.

Similarly, we have

$$\begin{aligned} f((1-\epsilon)n/e) &= \ln(1+(1-\epsilon)n/e) - \ln(n/e) - \frac{1}{1+(1-\epsilon)n/e} \\ &= \ln(1-\epsilon+e/n) - \frac{1}{1+(1-\epsilon)n/e} \\ &\le \ln(1-\epsilon/2) - \frac{1}{1+(1-\epsilon)n/e} \\ &\le -\epsilon/2 - o(n^{-1}) \\ &< 0 \end{aligned}$$

where the first inequality holds for large enough n since ln is an increasing function, the second inequality follows from $e^x \ge 1+x$, and the last inequality holds for large enough n.

Since $f(n/e) > 0$ and $f((1-\epsilon)n/e) < 0$, by the intermediate value theorem $f(z)$ has a root in the interval $((1-\epsilon)n/e + o(n), n/e + o(n))$. This is precisely the statement of this lemma.

Now we are in a position to prove the main theorem of this section.

Proof (Proof of Theorem 7) Fix $\epsilon > 0$. Consider MinRanking running on graph G_b. We introduce the following variables to denote the number of edges of certain types in a matching constructed by MinRanking:

- X_i denotes the number of $(S_{2,L}^{(i)}, S_{1,R})$ edges,
- Y_i denotes the number of $(S_{2,L}^{(i)}, S_{2,R})$ edges,
- Z denotes the number of $(S_{1,L}, S_{2,R})$ edges.

MinRanking first matches nodes in $S_{2,L}$ since these nodes have the minimum degree of $b+1$. Once MinRanking matches a node in some block $S_{2,L}^{(i)}$, it will continue matching the nodes in the same block until that block is exhausted. The algorithm then moves on to the next block. This continues for at least $b-1$ blocks. In the unlikely scenario that all b^2-b nodes of $S_{1,R}$ have been matched in this process, nodes from $S_{3,L}$ would have degree $b+1$. Therefore, after considering at least $b-1$ blocks from $S_{2,L}$ the algorithm might start matching nodes from other parts of L, but not sooner. The subgraph induced by $S_{2,L}^{(i)} \cup S_{2,R}^{(i)} \cup S_{1,R}^{(i)}$ is $H_{b,b}$, where $S_{1,R}^{(i)}$ consists of the nodes connected in parallel to $S_{2,L}^{(i)}$. Therefore, by Lemma 10 we have

$$b^2/e + o(b^2) \ge \mathbb{E}\left[\sum_{i=1}^{b} X_i\right] \ge (1-\epsilon)b^2/e + o(b^2) \tag{1}$$

The algorithm will match all the vertices from the first $b-1$ blocks of $S_{2,L}$. Therefore, we have

$$b^2 \ge \sum_{i=1}^{b} X_i + Y_i \ge (b-1)b. \tag{2}$$

Note that X_i also counts the number of nodes from $S_{2,R}^{(i)}$ available to be matched with $S_{1,L}$. Therefore, disregarding nodes from $S_{3,L}$ the number of nodes matched between $S_{1,L}$ and $S_{2,R}$ is at least $\left(\sum_{i=1}^{b} X_i\right) - 2b$, i.e., we have

$$\left(\sum_{i=1}^{b} X_i\right) \geq Z \geq \left(\sum_{i=1}^{b} X_i\right) - 2b. \tag{3}$$

Putting all this together, the expected size of the matching output by MinRanking is

$$\begin{aligned}
\mathbb{E}\left[Z + \sum_{i=1}^{b} X_i + Y_i\right] &\geq \mathbb{E}[Z] + (b-1)b \\
&\geq \mathbb{E}\left[\sum_{i=1}^{b} X_i\right] - 2b + (b-1)b \\
&\geq (1-\epsilon)b^2/e + b^2 + o(b^2) \\
&= b^2(1 + (1-\epsilon)/e) + o(b^2),
\end{aligned}$$

where the first inequality is by (2), the second inequality is by (3), and the last inequality is by (1). Similarly, we have

$$\mathbb{E}\left[Z + \sum_{i=1}^{b} X_i + Y_i\right] \leq b^2(1 + 1/e) + o(b^2).$$

Since the graph has a perfect matching of size $2b^2 + 2b$, the asymptotic approximation ratio of MinRanking on G_b is at least $1/2 + (1-\epsilon)/(2e)$ and at most $1/2 + 1/(2e)$. Since $\epsilon > 0$ is arbitrary, the theorem follows.

The expectation $\mathbb{E}_{\sigma,\pi}[\text{Ranking}(\sigma,\pi)]$ in the analysis of MinRanking is with respect to the joint distribution for the Ranking permutation π of the offline nodes and the distribution on σ induced by the tie breaking rule for the online nodes. The expectation can be written as $\mathbb{E}_{\sigma}[\mathbb{E}_{\pi}[\text{Ranking}(\sigma,\pi)|\sigma]]$, the expectation with respect to π conditioned on the instantiations of σ. By averaging this implies that there is some fixed setting of σ for which the inappropximation result holds. In this regard, our inapproximation for MinRanking is the analogue of Besser and Poloczek's priority analysis of the MinGreedy algorithm. Both results show that what seems to be a "well-motivated" deterministic (or random) ordering of the online vertices performs worse than a naive uniform random ordering of the online vertices.

6 Conclusion and Open Problems

We have considered a number of "online-based" algorithms for the maximum bipartite matching 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 MBM, in conjunction with the results in Poloczek et al. [53] 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 that in some way can be considered as expansions of the online model. In particular, 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"?

There are a number of more specific immediate questions that deserve consideration. We collect a few of them here.

Open Problem 1 *Does there exist a deterministic multi-pass linear time algorithm that matches or improves upon the* $1 - 1/e$ *approximation of the* Ranking *algorithm?*

Open Problem 2 *What is the exact approximation ratio achieved by* MinRanking*? What is the best achievable ratio by any randomized priority algorithm? Note that Pena and Borodin [51] show that every deterministic priority algorithm for MBM cannot asymptotically exceed the* $1/2$ *approximation that holds for any greedy algorithm.*

Open Problem 3 *Find a conceptually simple tie-breaking rule for the greedy algorithm that beats the $1 - 1/e$ approximation ratio in the known IID model.*

Open Problem 4 *Is there a randomized priority algorithm for MBM that achieves an approximation ratio better than that of* Ranking *in the ROM model?*

Open Problem 5 *Is there a simple deterministic multi-pass algorithm for the vertex weighted matching problem where the offline vertices are weighted? Note that Aggarwal et al. [1] show that a "perturbed" version of KVV achieves a $1 - 1/e$ approximation for this vertex weighted case.*

Open Problem 6 *What advice complexity (if any) is needed for a deterministic or randomized priority algorithm to improve upon the $1 - 1/e$ approximation of the randomized KVV algorithm? In addition, can such an algorithm improve upon the $.696$ approximation of* Ranking *in the ROM input model?*

Acknowledgements We thank the anonymous reviewers for the conference version of this paper as well as the journal version of this paper. Their detailed and constructive comments helped us to improve the presentation of results in this paper. We also thank Joan Boyar and Kim Larsen, who pointed out a confusing typo in the proof of the positive result for our multi-pass algorithm in an earlier version of the paper.

References

1. Aggarwal, G., Goel, G., Karande, C., Mehta, A.: Online vertex-weighted bipartite matching and single-bid budgeted allocations. In: Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, January 23-25, 2011, pp. 1253–1264 (2011)
2. Angelopoulos, S., Borodin, A.: On the power of priority algorithms for facility location and set cover. In: Proc. of APPROX, pp. 26–39 (2002)
3. Angelopoulos, S., Borodin, A.: Randomized priority algorithms. Theoretical Computer Science **411**(26), 2542 – 2558 (2010)
4. Aronson, J., Dyer, M., Frieze, A., Suen, S.: Randomized greedy matching. II. Random Struct. Algorithms **6**(1), 55–73 (1995)
5. Bahmani, B., Kapralov, M.: Improved bounds for online stochastic matching. In: Proc. of ESA, pp. 170–181 (2010)
6. Besser, B., Poloczek, M.: Erratum to: Greedy matching: Guarantees and limitations. Algorithmica pp. 1–4 (2017)
7. Besser, B., Poloczek, M.: Greedy matching: Guarantees and limitations. Algorithmica **77**(1), 201–234 (2017)
8. Birnbaum, B., Mathieu, C.: On-line bipartite matching made simple. SIGACT News **39**(1), 80–87 (2008)
9. Böckenhauer, H.J., Komm, D., Královič, R., Královič, R.: On the advice complexity of the k-server problem. J. of Comput. and System Sciences **86**, 159 – 170 (2017)
10. Borodin, A., Boyar, J., Larsen, K.S.: Priority Algorithms for Graph Optimization Problems, pp. 126–139. Springer Berlin Heidelberg (2005)
11. Borodin, A., Ivan, I., Ye, Y., Zimny, B.: On sum coloring and sum multi-coloring for restricted families of graphs. Theoretical Computer Science **418**, 1 – 13 (2012)
12. Borodin, A., Karavasilis, C., Pankratov, D.: An Experimental Study of Algorithms for Online Bipartite Matching. ArXiv e-prints (2018)
13. Borodin, A., Nielsen, M.N., Rackoff, C.: (incremental) priority algorithms. Algorithmica **37**(4), 295–326 (2003)
14. Boyar, J., Favrholdt, L.M., Kudahl, C., Larsen, K.S., Mikkelsen, J.W.: Online algorithms with advice: A survey. ACM Comput. Surv. **50**(2), 19:1–19:34 (2017)
15. Brubach, B., Sankararaman, K.A., Srinivasan, A., Xu, P.: New Algorithms, Better Bounds, and a Novel Model for Online Stochastic Matching. In: Proc. of ESA, pp. 24:1–24:16 (2016)
16. Brubach, B., Sankararaman, K.A., Srinivasan, A., Xu, P.: New algorithms, better bounds, and a novel model for online stochastic matching. In: Proc. of ESA, pp. 24:1–24:16 (2016)
17. Buchbinder, N., Feldman, M.: Deterministic algorithms for submodular maximization problems. In: Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016, pp. 392–403 (2016)
18. Chandra, B., Halldórsson, M.M.: Greedy local improvement and weighted set packing approximation. J. Algorithms **39**(2), 223–240 (2001)
19. Davis, S., Impagliazzo, R.: Models of greedy algorithms for graph problems. Algorithmica **54**(3), 269–317 (2009)
20. Dean, J., Ghemawat, S.: Mapreduce: simplified data processing on large clusters. Commun. ACM **51**(1), 107–113 (2008)
21. Devanur, N.R., Jain, K., Kleinberg, R.D.: Randomized primal-dual analysis of ranking for online bipartite matching. In: Proc. of SODA, pp. 101–107 (2013)
22. Duan, R., Pettie, S.: Linear-time approximation for maximum weight matching. J. ACM **61**(1), 1:1–1:23 (2014)
23. Dürr, C., Konrad, C., Renault, M.: On the Power of Advice and Randomization for Online Bipartite Matching. In: Proc. of ESA, pp. 37:1–37:16 (2016)
24. Eggert, S., Kliemann, L., Munstermann, P., Srivastav, A.: Bipartite matching in the semi-streaming model. Algorithmica **63**(1-2), 490–508 (2012)
25. Epstein, L., Levin, A., Mestre, J., Segev, D.: Improved approximation guarantees for weighted matching in the semi-streaming model. SIAM J. Discrete Math. **25**(3), 1251–1265 (2011)
26. Epstein, L., Levin, A., Segev, D., Weimann, O.: Improved bounds for online preemptive matching. In: Proc. of STACS, pp. 389–399 (2013)
27. Feigenbaum, J., Kannan, S., McGregor, A., Suri, S., Zhang, J.: On graph problems in a semi-streaming model. Theor. Comput. Sci. **348**(2-3), 207–216 (2005)

28. Feldman, J., Mehta, A., Mirrokni, V., Muthukrishnan, S.: Online stochastic matching: Beating 1-1/e. In: Proc. of FOCS, pp. 117–126 (2009)
29. Goel, A., Kapralov, M., Khanna, S.: 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, pp. 468–485 (2012)
30. Goel, G., Mehta, A.: Online budgeted matching in random input models with applications to adwords. In: Proc. of SODA, pp. 982–991 (2008)
31. Haeupler, B., Mirrokni, V.S., Zadimoghaddam, M.: Online stochastic weighted matching: Improved approximation algorithms. In: Proc. of WINE, pp. 170–181 (2011)
32. Halldórsson, M.M., Iwama, K., Miyazaki, S., Taketomi, S.: Online independent sets. Theor. Comput. Sci. **289**(2), 953–962 (2002)
33. Hopcroft, J.E., Karp, R.M.: An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. SIAM J. on Comput. **2**(4), 225–231 (1973)
34. Hosaagrahara, M., Sethu, H.: Degree-sequenced matching algorithms for input-queued switches. Telecommunications Systems **34**(37) (2007)
35. Jaillet, P., Lu, X.: Online stochastic matching: New algorithms with better bounds. Mathematics of Operations Research **39**(3), 624–646 (2014)
36. Kapralov, M.: Better bounds for matchings in the streaming model. In: Proc. of SODA, pp. 1679–1697 (2013)
37. Karande, C., Mehta, A., Tripathi, P.: Online bipartite matching with unknown distributions. STOC pp. 587–596 (2011)
38. Karp, R.M., Vazirani, U.V., Vazirani, V.V.: An optimal algorithm for on-line bipartite matching. In: Proc. of STOC, pp. 352–358 (1990)
39. Konrad, C., Magniez, F., Mathieu, C.: Maximum matching in semi-streaming with few passes. In: Proc. of APPROX and RANDOM, pp. 231–242 (2012)
40. Kurtz, T.G.: Solutions of ordinary differential equations as limits of pure jump markov processes. Journal of Applied Probability **7**(1), 49–58 (1970)
41. Lucier, B., Syrgkanis, V.: Greedy algorithms make efficient mechanisms. In: Proc. of Conference on Economics and Computation, EC, pp. 221–238 (2015)
42. Madry, A.: Navigating central path with electrical flows: From flows to matchings, and back. In: Proc. of FOCS, pp. 253–262 (2013)
43. Madry, A.: Computing maximum flow with augmenting electrical flows. In: Proc. of FOCS), pp. 593–602 (2016)
44. Magun, J.: Greedy matching algorithms: An experimental study. ACM Journal of Experimental Algorithmics **3**, 6 (1998)
45. Mahdian, M., Yan, Q.: Online bipartite matching with random arrivals: An approach based on strongly factor-revealing LPs. In: Proc. of STOC, pp. 597–606 (2011)
46. Manshadi, V.H., Gharan, S.O., Saberi, A.: Online stochastic matching: Online actions based on offline statistics. In: Proc. of SODA, pp. 1285–1294 (2011)
47. McGregor, A.: Graph sketching and streaming: New approaches for analyzing massive graphs. In: Proc. of Intern. Comput. Sci. Symp. in Russia, CSR, pp. 20–24 (2017)
48. Mehta, A.: Online matching and ad allocation. Foundations and Trends® in Theoretical Computer Science **8**(4), 265–368 (2013)
49. Mikkelsen, J.W.: Randomization can be as helpful as a glimpse of the future in online computation. In: 43rd International Colloquium on Automata, Languages, and Programming, ICALP, pp. 39:1–39:14 (2016)
50. Mucha, M., Sankowski, P.: Maximum matchings via gaussian elimination. In: Proc. of FOCS, pp. 248–255 (2004)
51. Pena, N., Borodin, A.: On the limitations of deterministic de-randomizations for online bipartite matching and max-sat. CoRR **abs/1608.03182** (2016). URL http://arxiv.org/abs/1608.03182
52. Poloczek, M.: Bounds on Greedy Algorithms for MAX SAT, pp. 37–48 (2011)
53. Poloczek, M., Schnitger, G., Williamson, D.P., Zuylen, A.V.: Greedy algorithms for the maximum satisfiability problem: Simple algorithms and inapproximability bounds. SICOMP **46**(3), 1029–1061 (2017)
54. Poloczek, M., Szegedy, M.: Randomized greedy algorithms for the maximum matching problem with new analysis. In: 53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012, pp. 708–717 (2012)
55. Poloczek, M., Williamson, D.P.: An experimental evaluation of fast approximation algorithms for the maximum satisfiability problem. In: Proc. of Intern. Symp. on Experimental Algorithms, SEA, pp. 246–261 (2016)
56. Tinhofer, G.: A probabilistic analysis of some greedy cardinality matching algorithms. Annals of Operations Research **1**(3), 239–254 (1984)
57. Wormald, N.C.: Differential equations for random processes and random graphs. Ann. Appl. Probab. **5**(4), 1217–1235 (1995)
58. Wormald, N.C.: The differential equation method for random graph processes and greedy algorithms. In: Lectures on approximation and randomized algorithms, pp. 73–155. PWN, Warsaw (1999)
59. Ye, Y., Borodin, A.: Priority algorithms for the subset-sum problem. J. Comb. Optim. **16**(3), 198–228 (2008)

A Relevant Algorithmic Models

We elaborate on the relavant models given in the Introduction.

Adversarial Online Model. In the online bipartite matching introduced by Karp, Vazirani, and Vazirani [38], nodes V are the offline nodes known to an algorithm beforehand, and nodes U arrive online in some adversarial order. When a node in U arrives, all its neighbors in V are revealed simultaneously. An online algorithm is required to make an irrevocable decision with regard to which neighbor (if any) the arrived node is matched. Any greedy algorithm (yielding a maximal matching) achieves a 1/2 approximation and Karp, Vazirani, and Vazirani showed that no deterministic algorithm can achieve an (asymptotic) approximation ratio better than 1/2 in the adversarial online model. They 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. Seventeen years after the publication of the Ranking algorithm a mistake was found in the analysis of the Ranking algorithm. The mistake was discovered independently by Krohn and Varadarajan and by Goel and Mehta in [30], and a correct proof was provided by Goel and Mehta. Since then many different and simplified proofs of the approximation ratio of Ranking have been given (see [30,8,21]). Thus, the one-pass adversarial online setting for MBM is now reasonably well understood.

Online Stochastic Models. The Feldman et al. [28] known IID distributional model reflex the observation that in applications of MBM to online advertising, one often knows some statistics about the upcoming queries (online nodes). Feldman et al. model this 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 a special case, they define the IID model with integral arrival rates where the probability distribution p is assumed to have the following property: for each type, the expected number of online nodes of that particular type appearing in a random instance is an integer. By repeating nodes, it is easy to see that the performance of an algorithm with respect to a uniform distribution p carries over to arbitrary integral rates. As before, an algorithm is required to make an irrevocable decision about which neighbor to match the newly arriving online node to. 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, as those are sampled in the IID fashion. Thus, the adversary is more restricted than in the adversarial online model. Feldman et al. [28] describe an algorithm beating the $1 - 1/e$ barrier and achieving approximation ratio ≈ 0.67 for the known IID model with integral arrival rates. This work was followed by a long line of work including [5,46,31,35,15]. So far, the best approximation ratio for arbitrary arrival rates is ≈ 0.706 due to Jaillet and Lu [35] and the best approximation for integral types .7299 is due to Brubach et al [16]. Other online stochastic input models have been studied; e.g., the Random Order Model (ROM), and the Unknown IID model. Karande et al [37] show that an online algorithm with respect to the unknown i.i.d. model (i.e., where the type graph is not known to the algorithm) can be veiwed as an algorithm in the ROM model (where the adversary creates the set of input items and the input order is a random permutation of those items). In the known IID model with integral types, the probability distribution p is assumed to have the following property: for each type, the expected number of online nodes of that particular type appearing in a random instance is an integer. It is easy to see that the performance of an algorithm with respect to a uniform distribution p carries over to arbitrary integral types. From now on, whenever we mention “known IID” we refer to known IID with a uniform distribution. example of a problem that is at the intersection of practice and theory.

Semi-streaming Model. The semi-streaming model (with edge inputs) was introduced by Feigenbaum et al. [27] and subsequently studied in [25,39,24]. In particular, Eggert et al. [24] 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. [29] give a *deterministic* $1 - 1/e$ approximation and Kapralov [36] proves that no semi-streaming algorithm can improve upon this ratio. (See also the recent survey by McGregor [47].) The difference between semi-streaming algorithms and online algorithms in the sense of competitive analysis is that 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. The Goel et al. result shows the power of deterministic semi-streaming over deterministic online algorithms. In some cases, streaming algorithms are designed so as to make results available at any time (after each input item) during the computation and hence some streaming algorithms can also be viewed both as a streaming algorithm and an online algorithm. Conversely, any online algorithm that restricts itself to $\tilde{O}(n)$ space can also be considered as a semi-streaming algorithm.

Priority Model. The priority model captures the idea that many offline greedy algorithms initially sort the input items, and then do a single pass over the items in the sorted order. More generally, priority algorithms can adaptively reorder so as to select the next input item to process. Many problems have been studied in the priority model [2,10,59,52,19,11,7]. The original deterministic priority model was extended to the randomized priority model in [3]. We shall use the term *fully randomized priority algorithm* to indicate that the ordering of the input items and the decisions for each item are both 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. [4] proved that an algorithm that picks a random vertex and matches it to a random available neighbor (if it exists) achieves approximation ratio at least $\frac{1}{2} + \frac{1}{400000}$ in general graphs. This is improved by Poloczek and Szegedy [54] to $\frac{1}{2} + \frac{1}{256}$. Besser and Poloczek [7] 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 [51] show that no deterministic (respectively, fully randomized) priority algorithm can achieve approximation ratio better than 1/2 (respectively 53/54) for the MBM problem. (See also [6] with respect to the difficulty of proving inapproximation results for all randomized priority algorithms.)

Advice Model. Dürr et al. show that for $\epsilon \in (0, 1/e)$, $\Theta_\epsilon(n)$ advice bits are necessary and sufficient for obtaining a $(1 - \epsilon)$ approximation[8]. where the high ϵ regime is due to Mikkelsen [49]. Dürr et al. also show that $O(\log n)$ advice bits are sufficient for a deterministic advice algorithm to guarantee a $1 - 1/e$ approximation ratio. This result is based on the randomization to advice transformation due to Böckenhauer et al. [9]. Thus there is an interesting sharp threshold phenomenon happening at approximation ratio $1 - 1/e$, where the required advice length jumps from $O(\log n)$ to $\Omega(n)$. 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

[8] More precisely, for $\epsilon \in (0, 1/e)$, the requirement for linear advice was proved by Mikkelsen [49]. Sufficiency is based on the multi-pass semi-streaming result of Eggert et al. [24]. Dürr et al. provide explicit bounds as to the dependence on ϵ; namely, for any $\epsilon > 0, O(\frac{n}{\epsilon^5})$ advice bits are sufficient, and for positive $\epsilon \leq 1/6$, $\Omega(\log(\frac{1}{\epsilon})n)$ advice bits are necessary to achieve a $(1 - \epsilon)$ approximation. The linear advice algorithm lacks the simplicity of the Dürr et al. [23] Category Advice algorithm.

construct $O(\log n)$ advice bits. More specifically, 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.

B Consistent Greedy Algorithms

Informally, the consistency condition says that if an online node u is matched with u^* in some run of the algorithm and u^* is available in another "similar" run of the algorithm, u should still be matched with u^* ("similar" means that neighbors of u in the second run form a subset of the neighbors of u in the first run). More formally:

Definition 4 Let ALG be a greedy algorithm. Fix an instance graph $\widehat{G}$, so that it is no longer a random variable. Let $\widehat{G}(\pi)$ denote an instance graph with online nodes presented in the order given by π. Let $N_\pi(u)$ denote the set of neighbors of u that are available to be matched when u is processed by ALG running on $\widehat{G}(\pi)$. We say that a greedy algorithm is *consistent or that it satisfies consistency conditions* if the following holds: $\forall \pi, \pi', u$ if $N_{\pi'}(u) \subseteq N_\pi(u)$ and u is matched with $u^* \in N_{\pi'}(u)$ by $\text{ALG}(\widehat{G}(\pi))$ then u is also matched with u^* by $\text{ALG}(\widehat{G}(\pi'))$.