Online Stochastic Matching with Edge Arrivals

Nick Gravin
ITCS, Shanghai University of Finance and Economics, China

Zhihao Gavin Tang
ITCS, Shanghai University of Finance and Economics, China

Kangning Wang
Department of Computer Science, Duke University, Durham, NC, USA

Abstract

Online bipartite matching with edge arrivals remained a major open question for a long time until a recent negative result by Gamlath et al., who showed that no online policy is better than the straightforward greedy algorithm, i.e., no online algorithm has a worst-case competitive ratio better than 0.5. In this work, we consider the bipartite matching problem with edge arrivals in a natural stochastic framework, i.e., Bayesian setting where each edge of the graph is independently realized according to a known probability distribution.

We focus on a natural class of prune & greedy online policies motivated by practical considerations from a multitude of online matching platforms. Any prune & greedy algorithm consists of two stages: first, it decreases the probabilities of some edges in the stochastic instance and then runs greedy algorithm on the pruned graph. We propose prune & greedy algorithms that are 0.552-competitive on the instances that can be pruned to a 2-regular stochastic bipartite graph, and 0.503-competitive on arbitrary stochastic bipartite graphs. The algorithms and our analysis significantly deviate from the prior work. We first obtain analytically manageable lower bound on the size of the matching, which leads to a non-linear optimization problem. We further reduce this problem to a continuous optimization with a constant number of parameters that can be solved using standard software tools.

2012 ACM Subject Classification Theory of computation → Online algorithms; Theory of computation → Approximation algorithms analysis; Theory of computation → Mathematical optimization; Mathematics of computing → Graph algorithms

Keywords and phrases online matching, graph algorithms, prophet inequality

Digital Object Identifier 10.4230/LIPIcs.ICALP.2021.74

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://arxiv.org/abs/1911.04686`

Funding This work is supported by Science and Technology Innovation 2030 – "New Generation of Artificial Intelligence" Major Project No.(2018AAA0100903), National Natural Science Foundation of China (NSFC) grant 61932002, Program for Innovative Research Team of Shanghai University of Finance and Economics (IRTSHUFE), Fundamental Research Funds for the Central Universities.
Zhihao Gavin Tang: supported by NSFC grant 61902233.
Kangning Wang: supported by NSF grant CCF-1637397 and ONR grant N00014-19-1-2268.

1 Introduction

Matching theory is a central area in combinatorial optimization with a big range of applications [28]. Many market models for jobs, commercial products, dating, healthcare, etc., rely on matching as a fundamental mathematical primitive. These examples often aim to describe environments that evolve in real time and thus are relevant to the area of online bipartite matching initiated by a seminal paper of Karp, Vazirani and Vazirani [26]. In this work they consider the one-sided vertex-arrival model within the competitive analysis framework, i.e., vertices only on one side of a bipartite graph appear online and each new vertex reveals all its

incident edges. The algorithm immediately and irrevocably decides to which vertex (if any) the new arrival is matched. They studied the worst-case performance of online algorithms and solved the problem optimally with an elegant $(1 - 1/e)$-competitive algorithm, named RANKING. Later, the proof of the result has been simplified by a series of papers [6, 19, 13].

The interest in matching models and online bipartite matching problems in particular has been on the rise since a decade ago due to emergence of the internet advertisement industry and online market platforms [32]. With the large amount of available data on many online platforms from the day-to-day user activities, more recent literature has shifted more towards *stochastic* models, also called Bayesian in the economically oriented work. In particular, Feldman, Mehta, Mirrokni and Muthukrishnan [16] proposed a stochastic model in which online vertices are drawn i.i.d. from a known distribution and improved[1] the competitive ratio of the classic result by Karp, Vazirani and Vazirani to 0.67. The competitive ratio has been further improved by a series of papers [3, 30, 24] to 0.706. Another line of work [25, 29] studied the model in which online vertices arrive in a random order and showed that the RANKING algorithm is 0.696-competitive.

The aforementioned results and other works, e.g., [33, 7, 12, 36, 18, 22, 23, 2], have made remarkable progress on different online matching settings with vertex arrivals, i.e., models where all incident edges of a new vertex are reported to the algorithm. However, more general arrival models are much less understood. E.g., one of the most natural and nonrestrictive extensions of online bipartite matching to the model where edges appear online and must be immediately matched or discarded was not known to have a competitive ratio better than the greedy algorithm for a long time. Only a recent negative result by Gamlath et al. [18] closed this tantalizing question showing that no online algorithm can be better than 0.5-competitive in the worst case. Algorithms with better performance are only known for quite special family of graphs, e.g., bounded-degree graphs [8] and forests [32, 8], or under strong assumptions on the edge arrival order, e.g., random arrival order [21].

It might seem that the edge-arrival model is too general to allow non-trivial theoretical results without strong assumptions on the instance. Thus it is not very surprising that practically motivated models do not usually consider online setting with edge arrivals. On the other hand, most of the specific applications posses additional structure and extra information that might allow to break the theoretical barrier. The edge-arrival online model besides pure theoretical interest and clean mathematical formulation, is indeed relevant to practical problems not unlike the examples we discuss below.

Practical Motivation: Edge Arrivals

Imagine any online matching platform for job search, property market, or even online dating. All these instances can be viewed as online matching processes in bipartite graphs. They also share a common trait that the realization of any particular edge is not instantaneous, often consumes significant effort and time from one or both sides of the potential match, and may exhibit complex concurrent behavior across different parties of the market. The platform can be thought of as an online matching algorithm, if it has any degree of control to intervene in the process of edge formation at any point.[2] However, the platform does not have enough power to control the order in which edges are realized. Hence, using arbitrary edge arrival order seems to be an appropriate modeling choice in these situations.

[1] Their result holds under the assumption that the expected number of vertices for each type is an integer.

[2] Even if the platform cannot directly prohibit an edge formation or disallow certain matches, it usually can affect outcome indirectly by restricting access/information exchange between certain pairs of agents, so that they never consider each other as potential matches.

Another notable feature of these instances is the vast amount of historical data accumulated over time. The data enables the platform to estimate the probability of a potential match between any pair of given agents. Thus the Bayesian (stochastic) approach widely adapted in economics seems to be another reasonable modeling choice. This raises the following natural question that to the best of our knowledge has not been considered before: *Is there an online matching algorithm for stochastic bipartite graphs with edge arrivals that is better than greedy?* This question is the main focus of our work. Let us first specify the model in more details.

Our Model: Edge Arrivals in Stochastic Graphs

We call our model *online stochastic matching with edge arrivals*. It is a relaxation of the standard edge-arrival model that performs on a random bipartite graph. In particular, we assume the input graph G is stochastic. That is, each edge e exists (is realized) in G independently with probability p_e and the probabilities $(p_e)_{e\in E(G)}$ are known to the online algorithm.[3] The algorithm observes a sequence of edges arriving online in a certain (unknown) order. Upon the arrival of an edge e, we observe the realization of e and if e exists, then the algorithm immediately and irrevocably decides whether to add e to the matching. We assume that the arrival order of the edges is chosen by an *oblivious* adversary, i.e., an adversary who does not observe the realization of the edges and algorithm's decisions, which is a standard assumption in the literature on online algorithms in stochastic settings (see, e.g., [27]). We compare the expected performance of our algorithm with the maximum matching in hindsight, i.e., the expected size of a maximum matching over the randomness of all edges.

1.1 Comparison with Other Stochastic Models

Our model is closely connected to two existing theoretical lines of works on stochastic bipartite matching and prophet inequality in algorithmic game theory. Below we compare our model with the most relevant results in each of these lines of works.

Stochastic Probing Model

It has the same ingredient as our model: the underlying stochastic graph. That is, the input is also a bipartite graph with the stochastic information on existence probability of every edge e. On the other hand, it is an offline model under the query-commit framework, i.e., the algorithm can check the existence of the edges in any order. However, if an edge exists, it has to be included into the solution. For this model, an adaptation of the RANKING algorithm by Karp, Vazirani and Vazirani is $(1-1/e)$-competitive. Costello, Tetali and Tripathi [11] provided a 0.573-approximation algorithm on general (non-bipartite) graphs and showed that no algorithm can have an approximation ratio larger than 0.898. Recently, Gamlath, Kale and Svensson [17] designed a $(1-1/e)$-approximation algorithm for the edge-weighted version of this problem.

[3] Note that some independence assumption across the edges is necessary. If we allow arbitrary probability distribution over the sets of realized edges, the model would be as difficult as the worst case online setting.

Prophet Inequality for Bipartite Matching

Consider a bipartite graph, where all edges have random values independently sampled from given probability distributions. Upon the arrival of an edge, we see the realization of its value and decide immediately whether to include this edge if possible in the matching. This model was originally proposed by Kleinberg and Weinberg [27] for a more general setting of intersection of k matroids. Gravin and Wang [20] studied explicitly the setting of bipartite matching and provided a $\frac{1}{3}$-approximation. Ezra et al. [15] later improved this ratio to 0.337. Our model can be viewed as an unweighted version of this prophet setting. Indeed, we assume that each edge has value either 0 or 1 and, hence, the probability distribution is a product of Bernoulli random variables summarized by existence probabilities $(p_e)_{e\in E(G)}$. Note that the weighted case is strictly harder than the unweighted one. Gravin and Wang [20] provided a 1/2.25 hardness result for the weighted setting while our goal is to design an online algorithm with a competitive ratio strictly better than 1/2. After all, the simple greedy algorithm achieves a competitive ratio of 1/2 for unweighted graphs.

1.2 Our Results and Techniques

We study a specific family of algorithms, named *Prune & Greedy*. The algorithm consists of two steps: (i) prune the graph by removing or decreasing probabilities of certain edges in G; (ii) greedily take every edge in the pruned instance. In particular, upon the arrival of an edge, we always drop it with certain probability so that its realization probability is consistent with the pruned graph.

We argue that the family of *Prune & Greedy* algorithms is of independent interest due to their practical relevance. Indeed, in those market applications we discussed above, the online platform often cannot prevent the matching between two parties (pair of vertices) once they realized their compatibility. But the platform usually possesses all the stochastic information about the graph and thus is fully capable of implementing pruning step by restricting information to its users. After that participants naturally implement greedy matching by exploring compatibilities with the other side of the graph exposed to them by the platform in an arbitrary order.

As our first result (Theorem 5 and Theorem 14), we identify a class of graphs on which greedy algorithm performs better than the worst-case competitive ratio of 1/2. We compare the size of the matching to the total number of vertices, a stronger benchmark than the expected size of maximum matching. As the pruning step naturally decreases the expected size of the maximum matching, the change of the benchmark is indeed necessary. Specifically, we find that on log-normalized[4] c-regular graphs with small $c = 2$ the greedy algorithm matches at least 0.552 vertices. This result immediately implies that if initial stochastic graph has a 2-regular bipartite spanning subgraph, then *Prune & Greedy* algorithm is 0.552-competitive.

Second, we propose a 0.503-competitive *Prune & Greedy* algorithm for any bipartite stochastic graph (Theorem 8). This result confirms that the edge-arrival model is theoretically interesting in the stochastic framework. A complementary hardness result (Theorem 15) shows that no online algorithm can be better than 2/3-competitive.

[4] For any vertex in a log-normalized c-regular graph, the probability that it has at least an adjacent edge is $1 - e^{-c}$. The definition is given in Section 3. Informally, a log-normalized c-regular graph is a c-regular graph where all edges have weights $\varepsilon \approx 0$.

Our techniques

We first build some intuition by analyzing the greedy algorithm on log-normalized c-regular graphs. One of the main challenges is that different events such as "edge e is matched", or "vertex u is matched" may have complex dependencies. This makes it very difficult to write the performance of the greedy algorithm in an explicit analytical form. We consider simpler to analyze events: "there exists a vertex u whose first realized edge is the edge (uv)", which guarantee that vertex v is matched at the end of the algorithm. This relaxation allows us to break the analysis into independent optimization problems per each vertex. We derive a guarantee $f(c)$ on the fraction of vertices matched by the greedy algorithm for any c-regular stochastic graph, where the function $f(c)$ has a single peak around $c = 2$ with $f(2) \approx 0.532$. I.e., we develop an analytically tractable relaxation on the performance of greedy that we later generalize to non-regular case. Interestingly, the greedy algorithm may perform worse on log-normalized c-regular graphs for larger c. In particular, greedy is not better than 0.5-competitive on c-regular graphs as $c \to \infty$.

However, this relaxation alone is not sufficient for the general case of non-regular graphs, since such analysis is not tailored in any way to the expected size of optimal matching. To this end, we consider an LP relaxation (an upper bound) on the expected optimal matching in stochastic graphs proposed in [17]. This LP gives a set of values $(x_e)_{e\in E(G)}$ with the objective $\sum_{e\in E(G)} x_e$ which satisfy a set of constraints that could be conveniently added to our optimization problem. Our analysis for regular graphs prompted us to the strategy of pruning each edge of the graph to $2 \cdot x_e$ so that the pruned graph is similar to a 2-regular graph. Unfortunately, this might not be a feasible operation when p_e (realization probability of e) is smaller than $2 \cdot x_e$. For these edges, it is then natural to keep their original existence probability. Our analysis can be similarly localized to an optimization problem for individual vertices, albeit the optimization becomes more complex. The main technical challenge is to solve an unwieldy optimization problem due to the "irregular" edges. Note that even a simpler optimization problem for c-regular graphs has a continuous optimal solution (i.e., is a limit of increasing discrete instances), which required computer assisted calculations to obtain the bound.

Finally, building on top of the relaxation we discussed above, we provide a more refined analysis for the case of 2-regular graphs. Namely, we consider second-order events that also witness the matching status of a vertex. We prove that the greedy algorithm is at least 0.552-competitive on 2-regular graphs, improving on the easier $f(2) \approx 0.532$ bound. We note that the same approach could in principle be extended to general *Prune & Greedy* algorithm for arbitrary graphs with optimization part still localizable to individual vertices. However, the optimization problem becomes too complicated to solve analytically. We leave it as an interesting open question to have a better analysis of the *Prune & Greedy* algorithms. On the positive side, the improved analysis for 2-regular graphs suggests that performance of *Prune & Greedy* algorithms should be noticeably better than what we proved in this paper.

1.3 Other Related Works

The edge-arrival setting is also studied under the free-disposal assumption, i.e., the algorithm is able to dispose of previously accepted edges. McGregor [31] gave a deterministic $\frac{1}{3+2\sqrt{2}} \approx 0.171$-competitive algorithm for weighted graphs. Varadaraja [35] proved the optimality of this result among deterministic algorithms. Epstein, Levin, Segev and Weimann [14] gave a $\frac{1}{5.356} \approx 0.186$-competitive randomized algorithm later and proved a hardness result of $\frac{1}{1+\ln 2} \approx 0.591$ for unweighted graphs. Recently, the hardness bound is improved to $2 - \sqrt{2} \approx 0.585$ by Huang et al. [23]. We remark that the question of designing an algorithm that beats 0.5-competitive remains open.

One of the earlier work on stochastic matching is due to Chen et al. [9]. They proposed stochastic model with edge probing motivated by real life matching applications such as kidney exchange. This model is more complex than the stochastic probing model we discussed before, since it has an additional constraint per each vertex v on how many times edges incident to v can be queried. Another difference is that a weaker benchmark than the optimal offline matching has to be used in this setting. Chen et al. developed a $\frac{1}{4}$-approximation algorithm. Bansal et al. [4] considered the weighted version and provided a $\frac{1}{3}$-approximation and a $\frac{1}{4}$-approximation for bipartite graphs and general graphs respectively. The ratio for general graphs was further improved to $\frac{1}{3.709}$ by Adamczyk, Grandoni and Mukherjee [1], and then to $\frac{1}{3.224}$ by Baveja et al. [5].

There has been prior work in the online matching literature on regular graphs [3, 25, 34, 21, 10]. However, the notion of stochastic regular graphs we defined is very different from the notion of standard regular graphs, the previous techniques are not directly applicable.

2 Preliminaries

The bipartite graph $G = (L, R, E)$ consists of left and right sides denoted respectively L and R. The graph G is a multigraph, i.e., E is a multiset that may have multiple parallel edges between the same pair of vertices. We use E_v to denote the multiset of edges incident to vertex v and E_{uv} to denote the multiset of edges connecting u and v. We consider the Bayesian model, where each edge $e \in E$ is realized with probability $p_e \in [0, 1]$, which is known in advance. The realizations of different edges are independent. We are interested in online matching algorithms with the objective of maximizing the expected size of the matching. We assume that all edges in E arrive one by one according to some fixed unknown order (i.e., oblivious adversarial order). Upon arrival of the edge e, the algorithm observes whether or not e is realized. If the edge exists, the algorithm immediately and irrevocably decides whether to include e into the matching; the algorithm does nothing, if the edge is not realized. We compare the performance of the algorithm with the performance of the optimal offline algorithm, also known as the *prophet*, who knows the realization of the whole graph in hindsight, i.e., $\mathsf{OPT} = \mathbf{E}[\text{size of maximum matching}]$.

A natural online matching strategy is the greedy algorithm: Take every available edge $e = (u, v)$ whenever both vertices u and v have not yet been matched. Obviously, the greedy algorithm is a 0.5-approximation, since it selects a *maximal* matching in all possible realizations of the graph, which is always a 0.5-approximation to the *maximum* matching.

Paper Roadmap

In Section 3, we introduce the notion of stochastic regular graphs and establish an analytical bound on the competitive ratio of the greedy algorithm on c-regular graphs. In Section 4, we design a Prune & Greedy algorithm that is 0.503-competitive for general inputs. Section 5 provides a more refined analysis of the greedy algorithm on 2-regular graphs. In Section 6, we give a simple impossibility result showing that no online algorithm can do better than $\frac{2}{3}$ of the expected optimum. Omitted proofs are in the full version.

3 Warm-up: Regular Graphs

A regular graph is a graph whose vertices have the same degree. But how do we define vertex degrees in a stochastic graph? One standard way is to use the expected vertex degree, i.e., $\sum_{e \in E_u} p_e$ for the degree of a vertex $u \in L$. However, the expectation alone does not

contain all the important information about a degree distribution. Consider for example a vertex a having only one incident edge (a, b) with $p_{(a,b)} = 1$ and a vertex u having 2 incident parallel edges $e = (u, v)$ with probability $p_e = 0.5$ for each $e \in E_u$. Both vertices a and u have the same expected degree, but while a always has exactly one incident edge, u gets no incident edges with 0.25 probability. On the other hand, u may have 2 incident edges in some realizations, which is almost the same for our purposes as having only a single incident edge.

A good way to reconcile this difference is to substitute each edge (u, v) by multiple parallel edges $e_i = (u, v)$ with small probabilities such that $p_{(u,v)}$ matches the probability that at least one of e_i edges exists. Alternatively, we can define a *log-normalized weight* for each edge e as $w_e \stackrel{\text{def}}{=\!=} -\ln(1 - p_e)$, i.e., given an input instance $G = (L, R, E)$ we construct a one-to-one correspondence between vectors of probabilities $\mathbf{p} = (p_e)_{e\in E}$ and vectors of log-normalized weights $\mathbf{w} = (w_e)_{e\in E}$. In particular, if we split an edge with log-normalized weight w_e into two edges $e_1 = e_2 = (u, v)$ with $w_{e_1} + w_{e_2} = w_e$, then the new instance gets only harder, i.e., any online algorithm for the new instance can be easily adapted to the original instance with the same or better performance. Indeed, notice that the probability that at least one of the edges e_1, e_2 exists equals $1 - e^{-w_{e_1}} \cdot e^{-w_{e_2}} = 1 - e^{-w_e}$, the probability that e exists, i.e., there is a probability coupling between the event that e exists with the event that at least one of e_1, e_2 exists. Then, we can substitute e in any arrival order with a pair of consecutive edges e_1 and e_2 and match e whenever the online algorithm matches e_1 or e_2 in the modified instance. Thus the log-normalized weight is the correct notion for us to do additive operations over the existence probabilities and leads to the following definition of the regular stochastic graph.

▶ **Definition 1.** *A graph G is a* log-normalized c-regular graph *if for every $v \in L \cup R$, $\sum_{e\in E_v} w_e = c$.*

We restrict our attention to log-normalized regular graphs in the remainder of this section. Our goal is to analyze the performance of Greedy on log-normalized c-regular graphs for a small constant c. Remarkably, it is not easy to give a precise answer and produce a tight worst-case estimate even for a specific value $c = 1$.

We first introduce a few short hand notations for the events that will be frequently used throughout the paper.

▶ **Definition 2.** *Fix an arbitrary edge arrival order σ and an edge $e \in E_{uv}$, define the following events:*

1. *$\exists e$: the event that e is realized.*
2. *$M_u(e)$: the event that u is matched right before e arrives.*
3. *$Q_u(e)$: the event that no edge of E_u is realized before e arrives. Let $q_u(e) \stackrel{\text{def}}{=\!=} \mathbf{Pr}[Q_u(e)]$.*
4. *$F_u(e) \stackrel{\text{def}}{=\!=} Q_u(e) \cap \exists e$: the event that e is the first realized edge of vertex u.*

The following lemma gives a lower bound on the matching probability of any vertex. This analytically tractable bound will allow us to reduce the global optimization for the competitive ratio of our algorithm to the local optimization per individual vertex. The lemma will also be useful for the general case, i.e., for not necessarily regular graphs, which we discuss in Section 4. To be consistent with the notations of the next section, let $x_e = \frac{w_e}{c}$ and $y_e = 1 - e^{-w_e}$. We have the property that $\sum_{e\in E_v} x_e = 1$ for every $v \in V$ and y_e equals the probability that e is realized ($\exists e$). We state the following lemma with the weaker condition of $\sum_{e\in E_v} x_e \leqslant 1$ for more general use in later sections.

▶ **Lemma 3.** *For any $v \in R$, if $\sum_{e\in E_v} x_e \leqslant 1$, then*

$$\mathbf{Pr}\left[v \text{ is matched}\right] \geqslant \mathbf{Pr}\left[\bigcup_{e=(u,v)\in E_v} F_u\left(e\right)\right] \geqslant \sum_{e=(u,v)\in E_v} x_e \cdot \left(1-\exp\left(-\frac{q_u(e)\cdot y_e}{x_e}\right)\right). \quad (1)$$

Proof. For each edge $e \in E_v$, consider the case when edge $e = (u, v)$ arrives and the event $F_u(e) = Q_u(e) \cap \exists e$ happens. At this moment, either v is already matched, or e will be included in the matching by Greedy. Therefore, whenever $\exists u \in L$ such that there is an edge $e \in E_{uv}$ making $F_u(e)$ happen, v is covered by Greedy.

Next, the events $\left\{\bigcup_{e\in E_{uv}} F_u(e)\right\}_{u\in L}$ are mutually independent, since (i) the event $\bigcup_{e\in E_{uv}} F_u(e)$ only depends on the random realization of the edges in E_u and (ii) $E_u \cap E_{u'} = \varnothing$ when $u \neq u'$.[5]

Lastly, $F_u(e_1) \cap F_u(e_2) = \varnothing$ for any $e_1, e_2 \in E_{uv}$, and thus $\mathbf{Pr}[\bigcup_{e\in E_{uv}} F_u(e)] = \sum_{e\in E_{uv}} q_u(e) \cdot y_e$. Putting the above observations together, we have

$$\begin{aligned}
\mathbf{Pr}\left[v \text{ is matched}\right] &\geqslant \mathbf{Pr}\left[\bigcup_{e\in E_v} F_u\left(e\right)\right] = 1-\mathbf{Pr}\left[\bigcap_{e\in E_v} \overline{F_u\left(e\right)}\right] = 1-\prod_u \mathbf{Pr}\left[\bigcap_{e\in E_{uv}} \overline{F_u\left(e\right)}\right] \\
&= 1-\prod_u\left(1-\sum_{e\in E_{uv}} q_u(e)\cdot y_e\right) \geqslant 1-\prod_u \exp\left(-\sum_{e\in E_{uv}} q_u(e)\cdot y_e\right) \\
&= 1-\exp\left(-\sum_{e\in E_v} q_u(e)\cdot y_e\right) \geqslant \sum_{e\in E_v} x_e\cdot\left(1-\exp\left(-\frac{q_u(e)\cdot y_e}{x_e}\right)\right),
\end{aligned}$$

where the second inequality follows from the fact that $1 - z \leqslant e^{-z}$ and the last inequality follows from Jensen's inequality and the concavity of function $1 - \exp(-z)$ (recalling $\sum_{e\in E_v} x_e \leqslant 1$). ◀

Thus, we may think of the quantity $x_e \cdot \left(1-\exp\left(-\frac{q_u(e)\cdot y_e}{x_e}\right)\right)$ as the contribution of edge e in the algorithm.[6] Observe that this contribution depends on the event $Q_u(e)$ for $u \in L$. We sum the (1) bound over all $v \in R$ and change the order of summations.

$$\begin{aligned}
\mathsf{ALG} = \sum_{v\in R}\mathbf{Pr}\left[v \text{ is matched}\right] &\geqslant \sum_{v\in R}\sum_{e=(u,v)\in E_v} x_e\cdot\left(1-\exp\left(-\frac{q_u(e)\cdot y_e}{x_e}\right)\right) \\
&= \sum_{u\in L}\sum_{e=(u,v)\in E_u} x_e\cdot\left(1-\exp\left(-\frac{q_u(e)\cdot y_e}{x_e}\right)\right). \quad (2)
\end{aligned}$$

▶ **Lemma 4.** *For all $u \in L$,*

$$\sum_{e\in E_u} x_e\cdot\left(1-\exp\left(-\frac{q_u(e)\cdot y_e}{x_e}\right)\right) \geqslant \int_0^1 \left(1-e^{-ce^{-cz}}\right)\mathrm{d}z.$$

Proof. Let u be any fixed vertex in L and $e_1, e_2, \ldots, e_k$ be the edges of E_u enumerated according to their arrival order. For notation simplicity, we use q_i, x_i and y_i to denote $q_u(e_i), x_{e_i}$ and y_{e_i} respectively. Then we have

$$Q_u\left(e_i\right) = \bigcap_{j<i}\overline{\exists e_j} = \bigcap_{j<i}\nexists e_j; \qquad q_i = \mathbf{Pr}\left[Q_u\left(e_i\right)\right] = \prod_{j<i}(1-y_j) = \prod_{j<i} e^{-c\cdot x_j} = e^{-c\cdot\sum_{j<i} x_j}.$$

[5] It is the only place where we use that G is bipartite. Indeed, our result can be generalized to triangle-free graphs.

[6] Note that this quantity is not necessarily a lower bound of the probability that edge e is matched.

Since $\int_0^{x_i} c \cdot e^{-cz} \mathrm{d}z = 1 - e^{-cx_i} = y_i$ and $1 - \exp(-q_i \cdot z)$ is a concave function of z , we can apply Jensen's inequality to get

$$1 - e^{-\frac{q_i \cdot y_i}{x_i}} = 1 - \exp\left(-q_i \cdot \frac{1}{x_i} \int_0^{x_i} c \cdot e^{-cz} \mathrm{d}z\right) \geqslant \frac{1}{x_i} \int_0^{x_i} \left(1 - \exp\left(-q_i \cdot c \cdot e^{-cz}\right)\right) \mathrm{d}z$$
$$= \frac{1}{x_i} \int_0^{x_i} \left(1 - \exp\left(-c \cdot e^{-c \cdot \sum_{j<i} x_j} \cdot e^{-cz}\right)\right) \mathrm{d}z = \frac{1}{x_i} \int_{\sum_{j<i} x_j}^{\sum_{j \leqslant i} x_j} \left(1 - e^{-ce^{-cz}}\right) \mathrm{d}z.$$

Summing this inequality over $i \in [k]$, we have

$$\sum_{e \in E_u} x_e \cdot \left(1 - \exp\left(-\frac{q_u(e) \cdot y_e}{x_e}\right)\right) = \sum_{i=1}^{k} x_i \cdot \left(1 - \exp\left(-\frac{q_i \cdot y_i}{x_i}\right)\right)$$
$$\geqslant \sum_{i=1}^{k} \int_{\sum_{j<i} x_j}^{\sum_{j \leqslant i} x_j} \left(1 - e^{-ce^{-cz}}\right) \mathrm{d}z = \int_0^1 \left(1 - e^{-ce^{-cz}}\right) \mathrm{d}z. \quad \blacktriangleleft$$

Let us denote the lower bound in the Lemma 4 as $h_1(c) \stackrel{\text{def}}{=\!=} \int_0^1 \left(1 - e^{-ce^{-cz}}\right) \mathrm{d}z$.

▶ **Theorem 5.** *The competitive ratio of* Greedy *on c-regular graphs is at least $h_1(c)$, which is at least* 0.532 *when $c = 2$.*

Proof. By equation (2) and Lemma 4, we have

$$\mathsf{ALG} \geqslant \sum_{u \in L} \sum_{e \in E_u} x_e \cdot \left(1 - \exp\left(-\frac{q_u(e) \cdot y_e}{x_e}\right)\right) \geqslant \sum_{u \in L} h_1(c).$$

This concludes the theorem by noticing that $|L|$ is an upper bound of OPT. $\blacktriangleleft$

▶ Remark. When $c = 2$, the competitive ratio is at least $h_1(c) \geqslant 0.532$. Notice that h_1 has a peak at $c \approx 2.1$, but it gets smaller again for $c > 3$ (see the full version for a plot) and our analysis gives relatively weak results for large c. One reason is because of the relaxation from Lemma 3. On the other hand, Greedy indeed does not perform well on c-regular graphs when c is large. In particular, Greedy is no better than 0.5-competitive on c-regular graphs when c goes to infinity.

▶ **Theorem 6.** *Greedy is at most* 0.5*-competitive on log-normalized c-regular graphs when $c \to \infty$.*

Proof. Consider the graph shown in Figure 1. We use $L_1 = \{u_i\}_{i=1}^{n+1}, R_1 = \{v_j\}_{j=1}^{n+1}, L_2 = \{u'_i\}_{i=1}^{n}$ and $R_2 = \{v'_j\}_{j=1}^{n}$ to denote the vertices in the graph. The edges are defined as the following:

1. For each $i \in [n+1]$, there is a (red solid) edge (u_i, v_i) with existence probability $1 - \varepsilon$.
2. For each pair of $(u, v) \in (L_2 \times R_1) \cup (L_1 \times R_2)$, there is a (green/blue dashed) edge (u, v) with existence probability $1 - \varepsilon$.

It is easy to verify the graph is log-normalized regular. When $\varepsilon \to 0$, with high probability, the graph admits a perfect matching with size $2n + 1$. On the other hand, consider the case when the red edges arrive first. With high probability, all these edges exist and Greedy matches $n + 1$ edges. This finishes the proof since $\frac{n+1}{2n+1} \to \frac{1}{2}$ when $n \to \infty$. $\blacktriangleleft$

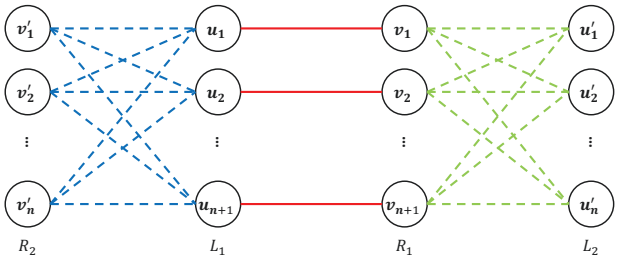

Figure 1 Hard instance for GREEDY on regular graphs.

4 Prune & Greedy: General Graphs

The fact that GREEDY beats half on log-normalized $c = 2$ regular graphs lends itself to the following natural two step adaptation for general graphs: (i) prune (remove or decrease probabilities of certain edges in G) such that the log-normalized degree of each vertex in the remaining graph is 2; (ii) greedily take every edge in the pruned instance G_c. Specifically, upon the arrival of an edge e, we first adjust its probability by dropping e so that its realization probability is consistent with e's log-normalized weight in G_c, then we match the realized edge if none of e's endpoints are currently matched. This approach would already yield the desired result for the dense graphs that can be pruned to the log-normalized 2-regular graph G_c. However, such a direct strategy fails for the graphs that have a few small degree vertices.

Before we proceed with the fix for the general graphs, let us take a closer look at the proof of Theorem 5. Note that in the theorem we actually compare our algorithm with a stronger benchmark, half the total number of vertices in G. The problem with such a benchmark, is that it may be too strong for any algorithm to approximate. To address this issue, we have to adjust our algorithm and analysis to handle low degree vertices. To this end, we can calculate x_e, the probability that e appears in the maximum matching of the random graph for every e, as the first step of our algorithm. By definition, $\mathsf{OPT} = \sum_{e\in E} x_e$ is the right benchmark to compare with. Alternatively, we can solve the following LP introduced by Gamlath, Kale and Svensson [17].[7]

$$\begin{aligned} \underset{(x_e \geqslant 0)_{e\in E}}{\text{maximize}} \quad & \sum_{e\in E} x_e \\ \text{subject to} \quad & \sum_{e\in F} x_e \leqslant 1 - \prod_{e\in F}(1-p_e), \quad \forall v \in L \cup R,\ \forall F \subseteq E_v. \end{aligned} \tag{3}$$

The constraints of the LP simply state that for each vertex v and subset $F \subseteq E_v$ of edges incident to v, the probability that an edge of F appears in the maximum matching is at most the probability that at least one edge of F is realized. Note that the value of each variable x_e in the LP (3) does not necessarily match the exact probability of e to appear in the maximum matching. However, $\sum_{e\in E} x_e$ still serves as a valid upper bound on OPT. As a matter of fact, our analysis works for either benchmark: the solution to LP (3), or for each x_e being the probability of e to appear in the optimal matching. We choose the LP (3) formulation in the description of the algorithm and the following analysis, since the LP optimal solution is a stronger benchmark and important constraints are explicitly stated in the LP.

[7] The LP is polynomial-time solvable. See [17] for the details.

A natural approach for general graphs would be to prune the graph according to the LP solution $(x_e)_{e\in E}$. To build some intuition let us consider what happens if we directly use x_es instead of p_es:

1. prune the graph by decreasing the probabilities of each edge from p_e to x_e,
2. run Greedy on the pruned instance.

Consider a special case of complete bipartite graph $K_{n,n}$ where each edge is realized with probability 1. The optimal solution to LP (3) is $x_e = \frac{1}{n}$ for all $e \in E$, as the maximum matching has size $n = \sum_{e\in E} x_e$. As $-\ln(1-x) \approx x$ when x is small, we effectively run Greedy on log-normalized 1-regular graph after pruning $K_{n,n}$. Theorem 3 from previous section gives $h_1(1) \approx 0.459 < 0.5$ in this case. Moreover, a simple computer aided simulation suggests that Greedy matches no more than 0.5 fraction of all vertices in this case. In this simulation we consider a regular complete graph G with $|L| = |R| = n$, where each edge has probability $\frac{1}{n}$; the edges arrive in random (uniformly distributed) order. In particular, we run 10^5 number of trials for $n = 3000$ and the ratio between ALG and n equals 0.50002. See the full version of this paper for more details of our experiment.

This means that pruning probabilities directly to x_e is too much and we need a more conservative pruning step. In particular, Theorem 5 suggests to prune the graph so that the log-normalized weight of edge e becomes $c \cdot x_e$. On the other hand, for some edges, p_e can be as small as x_e, in which case we have a cap on the existence probability. For those edges, it is reasonable to keep the existence probability as the original graph. Formally, we consider the following algorithm.

Algorithm 1 Prune & Greedy.

1: Solve LP (3) and let $\{x_e\}_{e\in E}$ be the optimal solution.
2: Prune the graph by decreasing the probabilities of each edge from p_e to $y_e = \min(p_e, 1 - e^{-c\cdot x_e})$.
3: Run Greedy on the pruned instance.

In an easy case when $y_e = 1 - e^{-c\cdot x_e}$ for all edges $e \in E$, we can adapt our analysis for c-regular graphs with a similar performance guarantee. On the other hand, if $y_e = x_e$ for all edges, then the algorithm might not be better than 0.5-competitive according to the previous discussion. However, the constraints from LP (3) guarantee that this cannot happen for all e.

Note that Lemma 3 and equation (2) apply to our Prune & Greedy algorithm with the $\{x_e, y_e\}$ defined in this section. We shall prove Lemma 7 an analog of Lemma 4 to conclude Theorem 8, which is the main result of this section.).

▶ **Lemma 7.** *For all $u \in L$, when $c = 1.7$,*

$$\sum_{e\in E_u} x_e \cdot \left(1 - \exp\left(-\frac{q_u(e)\cdot y_e}{x_e}\right)\right) \geqslant 0.503 \cdot \sum_{e\in E_u} x_e$$

▶ **Theorem 8.** *Prune & Greedy is 0.503-competitive when $c = 1.7$.*

Proof. We write the lower bound on the performance of Greedy using Lemma 3.

$$\begin{aligned}
\mathsf{ALG} &\geqslant \sum_{u\in L}\sum_{e\in E_u} x_e \cdot \left(1 - \exp\left(-\frac{q_u(e)\cdot y_e}{x_e}\right)\right) && \text{(by Equation (2))}\\
&\geqslant \sum_{u\in L} 0.503 \cdot \sum_{e\in E_u} x_e = 0.503 \cdot \sum_{e\in E} x_e \geqslant 0.503 \cdot \mathsf{OPT}. && \text{(by Lemma 7)}
\end{aligned}$$

◀

4.1 Proof of Lemma 7

The proof of Lemma 7 is highly technical and requires us to solve a rather non-trivial optimization. Let $\{e_1, e_2, \cdots, e_k\}$ be all edges incident to u and enumerated in their arrival order. To simplify notations, let p_i, x_i, y_i and q_i denote $p_{e_i}, x_{e_i}, y_{e_i}$ and $q_u(e_i)$ respectively.

The optimization problem (4) captures the ratio that we want to study.

$$\min_{(p_i \leqslant 1),(x_i \geqslant 0)} \sum_{i=1}^{k} x_i \cdot \left(1 - e^{-\frac{q_i \cdot y_i}{x_i}}\right) \Big/ \sum_{i=1}^{k} x_i \tag{4}$$
$$\text{s.t. } y_i = \min(p_i, 1 - e^{-c \cdot x_i}), \quad \forall i \in [k]$$
$$q_i = \prod_{j<i}(1 - y_j), \quad \forall i \in [k]$$
$$\sum_{i \in S} x_i \leqslant 1 - \prod_{i \in S}(1 - p_i), \ \forall S \subseteq [k]$$

$$\min_{(p_i \leqslant 1),x \geqslant 0} \sum_{i=1}^{k} x \cdot \left(1 - e^{-\frac{q_i \cdot y_i}{x}}\right) \Big/ (kx) \tag{5}$$
$$\text{s.t. } y_i = \min(p_i, 1 - e^{-cx}), \quad \forall i \in [k]$$
$$q_i = \prod_{j<i}(1 - y_j), \quad \forall i \in [k]$$
$$|S| \cdot x \leqslant 1 - \prod_{i \in S}(1 - p_i), \ \forall S \subseteq [k]$$

The first family of constraints in (4) comes from the design of our algorithm. The second family of constraints characterizes the probability that u has no realized edge before e_i. The last family of constraints follows from LP (3).

We first decrease the value of optimization (4) by increasing the size k of the instance and get a simpler optimization problem (5). The fact that we can construct more regular instance with all $x_i = x$ and smaller or equal objective value follows from the "subdivision" Lemma 9 below.

▶ **Lemma 9.** *Let $(x_j, p_j)_{j \in [k]}$ be any feasible solution to (4). Let e_i be any edge $i \in [k]$ which we subdivide into two consecutive parallel edges e' and e''. Then there is a feasible solution to the new instance of (4) with the same $(x_j, p_j)_{j \neq i}$ and $x_{e'} + x_{e''} = x_i$, and smaller objective value. Moreover, $x_{e'} \geqslant 0$ and $x_{e''} \geqslant 0$ can be set to have any values subject to $x_{e'} + x_{e''} = x_i$.*

Proof. We fix feasible solution $(x_j, p_j)_{j \in [k]}$ to (4), edge e_i, and particular $x_{e'} \geqslant 0$ and $x_{e''} \geqslant 0$ such that $x_{e'} + x_{e''} = x_i$. Let $\beta = \frac{x_{e'}}{x_i}$. Note that $0 \leqslant \beta \leqslant 1$. We need to define $p_{e'}$,$p_{e''}$. Consider two cases depending on the value of y_i:

Case 1. If $y_i = 1 - e^{-c \cdot x_i}$, then we let $p_{e'} = p_{e''} = 1$.

Case 2. If $y_i = p_i$, then we let $p_{e'} = 1 - (1 - p_i)^{\beta}$ and $p_{e''} = 1 - (1 - p_i)^{1-\beta}$.

In the first case, $y_{e'} = 1 - e^{-cx_{e'}}$ and $y_{e''} = 1 - e^{-cx_{e''}}$. In the second case, we verify that

$$p_{e'} \leqslant 1 - e^{-cx_{e'}} \iff 1 - (1 - p_i)^{\beta} \leqslant 1 - e^{-cx_{e'}} \iff p_i \leqslant 1 - e^{-cx_i}.$$

Similarly, $p_{e''} \leqslant 1 - e^{-cx_{e''}}$ and we have that $y_{e'} = p_{e'}$ and $y_{e''} = p_{e''}$. In both cases, we have that $(1 - y_{e'})(1 - y_{e''}) = 1 - y_i$. Consequently, the value of q_j does not change by our subdivision for all $j \neq i$. Next, we examine the change to the numerator of the objective function.

$$
\begin{aligned}
&x_{e'} \cdot \left(1 - e^{-q_i \cdot \frac{y_{e'}}{x_{e'}}}\right) + x_{e''} \cdot \left(1 - e^{-q_i \cdot \frac{(1-y_{e'})y_{e''}}{x_{e''}}}\right) - x_i \cdot \left(1 - e^{-q_i \cdot \frac{y_i}{x_i}}\right) \\
=&x_i \cdot \left(\frac{x_{e'}}{x_i} \cdot \left(1 - e^{-q_i \cdot \frac{y_{e'}}{x_{e'}}}\right) + \frac{x_{e''}}{x_i} \cdot \left(1 - e^{-q_i \cdot \frac{(1-y_{e'})y_{e''}}{x_{e''}}}\right)\right) - x_i \cdot \left(1 - e^{-q_i \cdot \frac{y_i}{x_i}}\right) \\
\leqslant&x_i \cdot \left(1 - e^{-\frac{q_i}{x_i} \cdot (y_{e'} + (1-y_{e'})y_{e''})}\right) - x_i \cdot \left(1 - e^{-q_i \cdot \frac{y_i}{x_i}}\right) = 0,
\end{aligned}
$$

where the inequality holds by Jensen's inequality for the concave function $1 - e^{-x}$, and the last equality is due to the fact that $(1-y_{e'})(1-y_{e''}) = 1-y_i$. Thus, the subdivision decreases the objective function. We are left to verify that all inequality constraints in (4) for $S \subseteq [k]$ are satisfied.

First, we consider **Case 1**, when $y_i = 1 - e^{-c \cdot x_i}$. We have $p_{e'} = p_{e''} = 1$. Then

- if $e', e'' \notin S$, the constraint trivially holds (none of x_j, p_j change);
- if $e' \in S$ or $e'' \in S$, the right-hand side of the constraint becomes 1.

Next, in **Case 2**, $y_i = p_i$, $y_{e'} = p_{e'}$, $y_{e''} = p_{e''}$, and $(1 - y_{e'})(1 - y_{e''}) = 1 - y_i$. Then

- if $e', e'' \notin S$, the constraint still holds (none of x_j, p_j change);
- if both $e', e'' \in S$, the inequality holds since $x_{e'} + x_{e''} = x_i$ and $(1 - p_{e'})(1 - p_{e''}) = 1 - p_i$;
- if exactly one of $e', e'' \in S$, we may assume w.l.o.g. that $e' \in S, e'' \notin S$ (as the other case $e'' \in S, e' \notin S$ is symmetric). Let $T \subseteq [k] - \{i\}$ be any set of indexes. Then

$$
\begin{aligned}
&1 - (1 - p_{e'}) \prod_{j \in T}(1 - p_j) = 1 - (1 - p_i)^{\beta} \prod_{j \in T}(1 - p_j) \geqslant 1 - (1 - \beta p_i) \prod_{j \in T}(1 - p_j) \\
&= \beta \left(1 - (1 - p_i) \prod_{j \in T}(1 - p_j)\right) + (1 - \beta) \left(1 - \prod_{j \in T}(1 - p_j)\right) \\
&\qquad \geqslant \beta \left(x_i + \sum_{j \in T} x_j\right) + (1 - \beta) \sum_{j \in T} x_j = \beta x_i + \sum_{j \in T} x_j = x_{e'} + \sum_{j \in T} x_j,
\end{aligned}
$$

where to get the first inequality we used the fact $(1 - x)^{\beta} \leqslant 1 - x \cdot \beta$, for any $x > -1$, $0 \leqslant \beta \leqslant 1$; the second inequality holds due to the original constraints in (4) for $S = T \cup \{i\}$ and $S = T$.

This concludes our proof. ◀

To get (5), we can start with the optimal solution to (4) for any given k, then apply multiple times Lemma 9 to every edge e_i, $i \in [k]$ getting an instance with $k' \gg k$ edges and a feasible solution with the same value, where almost all $x_j = x$ and at most k edges have $x_e < x$ (x may depend on k'). Finally, we can remove all edges with $x_e < x$, keep the rest x_j and p_j untouched and redefine (q_j) according to the recurrent formula. The impact of the change to q_j's before the removal of edges $x_e < x$ can be made vanishingly small as $k' \to \infty$. At the end, we get a feasible solution to (4) of the form (5) (for bigger k) with almost the same value as the optimum of (4) for the initial k. Thus we can analyze (5) without loss of generality instead of (4).

We prove the following lemma that describes the optimal solution to problem (5).

▶ **Lemma 10.** *For an optimal solution to* (5)*: (i)* $(y_i)_{i \in [k]}$ *are decreasing; (ii)* $\exists$ *cut-off point* $\ell \in [k]$ *such that* $y_i = cx$ *for* $i \leqslant \ell$ *and* $y_i = p_i$ *for* $i > \ell$*; (iii) constraints* $|S| \cdot x \leqslant 1 - \prod_{i \in S}(1 - p_i)$ *are tight for all* $S = [j..k]$*, where* $j > \ell$*.*

Proof. We prove the three statements sequentially. Let (p_i) be the optimal solution. If $y_i < y_{i+1}$ for an $i \in [k]$. Consider swapping p_i and p_{i+1} in the instance and the other $(p_j)_{j\neq i,i+1}$ remain the same. Note that the last family of constraints are preserved since the constraints are invariant under any permutation of p_j's. It is easy to see that q_j's are not changed for all $j \neq i+1$ and $q'_{i+1} = q_i(1-y_{i+1})$. Moreover, $y'_i = y_{i+1}$ and $y'_{i+1} = y_i$. Therefore, to prove that the swap decreases the objective, it suffices to show

$$1 - e^{-\frac{q_i \cdot y_i}{x}} + 1 - e^{-\frac{q_i(1-y_i)\cdot y_{i+1}}{x}} < 1 - e^{-\frac{q_i \cdot y_{i+1}}{x}} + 1 - e^{-\frac{q_i(1-y_{i+1})\cdot y_i}{x}}.$$

Observe that $q_i y_{i+1} > q_i y_i$, $q_i y_{i+1} > q_i(1-y_i)y_{i+1}$ and $q_i y_i + q_i(1-y_i)y_{i+1} = q_i y_{i+1} + q_i(1-y_{i+1})y_i$. The above inequality is true due to the convexity of the function $\exp(-z)$. A contradiction that concludes the proof of (i), the monotonicity of y_i's.

The second statement follows immediately from (i) according to our definition of $y_i = \min(p_i, cx)$ which is a monotone function with respect to p_i.[8] Let ℓ be the cut-off point such that $y_i = cx$ for $i \leqslant \ell$ and $y_i = p_i$ for $i > \ell$.

We are left to prove (iii). Given the statement (ii), we safely assume that $p_i = 1$ for all $i \leqslant \ell$ since this would not affect all y_i, q_i's and only trivialize the last family of constraints when $S \cap [\ell] \neq \varnothing$, since in this case the right-hand side of the constraint equals 1. Since $p_i = 1$ for $i \in [\ell]$ and $1 \geqslant p_i = y_i$ for $i > \ell$, we can assume that $(p_i)_{i\in[k]}$ are decreasing as well.

Given the monotonicity of p_i's, we note that "critical" inequality constraints $|S| \cdot x \leqslant 1 - \prod_{i\in S}(1-p_i)$ are those where $S = \{j, j+1, \cdots, k\}$, i.e., the remaining (non-critical) inequality constrains for other sets S are automatically satisfied, if the constraints for $S = \{j, j+1, \cdots, k\}$ hold. Indeed, when restricting to S with a fixed cardinality s, the left-hand side of each constraint is the same $|S| \cdot x$, while the right-hand side is minimized when S consists of the s smallest p_i, i.e., $\{p_j, p_{j+1} \ldots, p_k\}$. We are going to prove (iii), that the critical constraints for $j > \ell$ are tight.

Now, suppose to the contrary that a critical constraint is not tight for an $S = \{\bar{i}, \bar{i}+1, \ldots, k\}$ for $\bar{i} > \ell$, while all critical constraints for each $S = \{i, i+1, \ldots, k\}$ where $i > \bar{i}$ are tight (if $\bar{i} = k$, we don't require any constraints to be tight). We first consider a non-degenerate case when $1 > p_{\bar{i}-1}$, which also means that $\bar{i} - 1 > \ell$ (otherwise $y_{\bar{i}-1} = cx$ and we would set $p_{\bar{i}-1} = 1$). Before that we prove the following fact.

▷ Claim 11. If $1 > p_i = p_{i+1}$ for $i \in (\ell..k)$, then inequality $|S| \cdot x < 1 - \prod_{j\in S}(1-p_j)$ for $S = \{i+1, \ldots, k\}$ is strict.

Proof. Suppose to the contrary that the inequality is an equality, that is

$$(1-p_{i+1}) = \frac{\prod_{j>i}(1-p_j)}{\prod_{j>i+1}(1-p_j)} = \frac{1-(k-i)x}{\prod_{t>i+1}(1-p_j)}.$$

The inequality constraint for $S = \{i, \ldots, k\}$ gives:

$$(1-p_i)(1-p_{i+1}) = \frac{\prod_{j\geqslant i}(1-p_j)}{\prod_{j>i+1}(1-p_j)} \leqslant \frac{1-(k-i+1)x}{\prod_{j>i+1}(1-p_j)}.$$

[8] Notice that $cx \approx 1 - e^{-cx}$ when $x \approx 0$ in (5).

Putting the two equations together and by the assumption that $p_i = p_{i+1}$, we have

$$\left(\frac{1-(k-i)x}{\prod_{j>i+1}(1-p_j)}\right)^2 \leqslant \frac{1-(k-i+1)x}{\prod_{j>i+1}(1-p_j)}$$
$$\Longrightarrow \frac{(1-(k-i)x)^2}{(1-(k-i+1)x)} \leqslant \prod_{j>i+1}(1-p_j) \leqslant 1-(k-i-1)x,$$

where the last inequality follows from the constraint for $S = \{i+2, \dots, k\}$ (if $i+2 > k$, the inequality still holds, as $i = k-1$, $1-(k-i-1)x = 1$, and $\prod_{j>i+1}(1-p_j) = 1$).

Thus $(1-(k-i+1)x)(1-(k-i-1)x) = (1-(k-i)x)^2 - x^2 \geqslant (1-(k-i)x)^2$, a contradiction. ◁

Case 1 ($1 > p_{\bar{i}-1}$). We will get a contradiction by providing an instance with a strictly smaller objective's value. Let $\underline{i}$ be the smallest index so that $p_{\underline{i}} = p_{\underline{i}+1} = \cdots = p_{\bar{i}-1}$. So $p_{\underline{i}-1} > p_{\underline{i}}$, if $\underline{i} > 1$. Recall that we consider the case $1 > p_{\bar{i}-1} = p_{\underline{i}}$ and thus $\ell < \underline{i}$ (otherwise $y_{\underline{i}} = cx$ and we should have set $p_{\underline{i}} = 1$). By Claim 11, each inequality in (5) for $S = \{i, i+1, \dots, k\}$ with $\underline{i}+1 \leqslant i \leqslant \bar{i}-1$ must be strict. On the other hand, a contra-positive statement to Claim 11 gives us that $p_{\bar{i}}$ cannot be equal to $p_{\bar{i}+1}$ (if $\bar{i} = k$, this also is true). Thus $p_{\bar{i}} > p_{\bar{i}+1}$ (if $\bar{i} < k$). If $\bar{i} = k$, then $p_{\bar{i}} > 0$ (otherwise, we can decrease k in (5)).

We consider the following modification $(\mathbf{p}', x)$ of (5)'s feasible solution: slightly increase $p_{\underline{i}}$ and decrease $p_{\bar{i}}$ so that $(1-p_{\underline{i}})(1-p_{\bar{i}})$ remains the same; all other p_i for $i \neq \underline{i}, \bar{i}$ and x are the same in $(\mathbf{p}', x)$ and original optimum $(\mathbf{p}, x)$; $\mathbf{y}', \mathbf{q}'$ are redefined according to the formula in (5). Note that we can always do such modification when $1 > p_{\underline{i}} > p_{\bar{i}} > 0$.

For any sufficiently small such perturbation of $p_{\bar{i}}$ and $p_{\underline{i}}$, $(p'_i)_{i\in[k]}$ remain monotone and all constraints in (5) are satisfied. Indeed, we only need to check the critical constraints in (5) for monotone $\mathbf{p}'$: $\mathbf{p}'$ and $\mathbf{p}$ are the same for $S = \{i, i+1, \dots, k\}$ for $i \in (\bar{i}..k]$; all inequalities for $S = \{i, i+1, \dots, k\}$ where $i \in [\underline{i}+1, \bar{i}]$ are strict and, therefore, for sufficiently small perturbation of $p_{\bar{i}}$ and $p_{\underline{i}}$ they still hold; for $S = \{i, i+1, \dots, k\}$ where $i \in (\ell..\underline{i}]$, the right-hand side of each critical constraint does not change, because $(1-p_{\bar{i}})(1-p_{\underline{i}}) = (1-p'_{\bar{i}})(1-p'_{\underline{i}})$.

Now we examine the changes to q_i. Observe that each $q'_i = q_i$ and $y'_i = y_i$ for any $i < \underline{i}$, as $(p'_i)_{i<\underline{i}}$ and $(p_i)_{i<\underline{i}}$ are the same. For $i \geqslant \bar{i}$, we also have $q'_i = q_i$ and $y'_i = y_i$ since $(1-y_{\underline{i}})(1-y_{\bar{i}}) = (1-y'_{\underline{i}})(1-y'_{\bar{i}})$. Moreover, we notice that

$$\sum_{i\in[\underline{i}..\bar{i}]} q_i y_i = q_{\underline{i}}\left(1 - \prod_{i\in[\underline{i}..\bar{i}]}(1-y_i)\right) = \sum_{i\in[\underline{i}..\bar{i}]} q'_i y'_i.$$

In the interval $i \in [\underline{i}, \bar{i}]$, we notice that by increasing $p_{\underline{i}}$ and decreasing $p_{\bar{i}}$ we increase $q_{\underline{i}} y_{\underline{i}}$ and decrease each $q_i y_i$ for $i \in (\underline{i}..\bar{i})$, since each q_i deceases. Moreover,

$$q'_{\bar{i}} y'_{\bar{i}} = \left[\prod_{\substack{j<\bar{i},\\ j\neq\underline{i}}}(1-y_j)\right]\cdot(1-y'_{\underline{i}})\cdot y'_{\bar{i}} = \left[\prod_{\substack{j<\bar{i},\\ j\neq\underline{i}}}(1-y_j)\right]\cdot(1-y'_{\underline{i}}-(1-y'_{\bar{i}})(1-y'_{\underline{i}}))$$
$$< \left[\prod_{\substack{j<\bar{i},\\ j\neq\underline{i}}}(1-y_j)\right]\cdot(1-y_{\underline{i}}-(1-y_{\bar{i}})(1-y_{\underline{i}})) = q_{\bar{i}} y_{\bar{i}},$$

since $y'_{\underline{i}} > y_{\underline{i}}$ while $(1-y'_{\underline{i}})(1-y'_{\bar{i}}) = (1-y_{\underline{i}})(1-y_{\bar{i}})$. Hence, due to convexity of the function $\exp(-z)$, we conclude that the objective $\sum_i(1-\exp(-\frac{q_i y_i}{x}))$ decreases when we substitute $(\mathbf{q},\mathbf{y})$ with $(\mathbf{q}',\mathbf{y}')$. Indeed, $q_{\underline{i}}y_{\underline{i}}$, the largest number among $\{q_i y_i\}_{i=\underline{i}}^{\bar{i}}$, increases, while all other affected $q_i y_i$ in $[\underline{i},\bar{i}]$ decrease.

Case 2 ($p_{\bar{i}-1} = 1$). Now we consider a degenerate case when $p_{\bar{i}-1} = 1$. In this case, $p_{\bar{i}}$ only appears in the critical constraint for $S = \{\bar{i}, \bar{i}+1, \dots, k\}$, which we assume to be not tight. Note that $p_{\bar{i}} > p_{\bar{i}+1}$ if $\bar{i} < k$ by Claim 11 and also that $p_{\bar{i}} > 0$ if $\bar{i} = k$ (otherwise, we can decrease k in (5)). Thus, sightly decreasing $p_{\bar{i}}$ shall not violate any constraint. Furthermore, since $\bar{i} > \ell$, $p_{\bar{i}} < cx$, we can also slightly increase $p_{\bar{i}}$ without violating any constraints. Now, we fix all p_i for $i \neq \bar{i}$ and consider $y_{\bar{i}} = p_{\bar{i}}$ as a locally free variable that we can slight increase or decrease. We study the objective of (5) as a function of $y_{\bar{i}} = p_{\bar{i}}$.

Observe that any change to $y_{\bar{i}}$ only affects the terms $(1-e^{-\frac{q_i y_i}{x}})$ for $i \geqslant \bar{i}$. Furthermore, for $i = \bar{i}$, $(1-e^{-\frac{q_{\bar{i}} y_{\bar{i}}}{x}})$ is a strictly concave function of $y_{\bar{i}}$; and for $i > \bar{i}$

$$1-\exp\left(-\frac{q_i y_i}{x}\right) = 1-\exp\left(-(1-y_{\bar{i}})\cdot\frac{\prod_{j<i, j\neq\bar{i}}(1-y_j)\cdot y_i}{x}\right)$$

is also a concave function of $y_{\bar{i}}$.

Thus, the objective function of (5) is a strictly concave function of $y_{\bar{i}} = p_{\bar{i}}$, at least in some neighborhood of $p_{\bar{i}}$. Note that the minimum of a strictly concave function is always achieved on the boundary of its domain. Therefore, some small perturbation of $p_{\bar{i}}$ and consequently $y_{\bar{i}} = p_{\bar{i}}$ (either slightly increase or decrease $p_{\bar{i}}$ such that all constraint in (5) are still satisfied) would strictly decrease the objective. This contradicts the optimality of $\mathbf{p}$. ◀

Now we can write explicit formula for p_i for $i > \ell$. By (iii) of Lemma 10, we have $\prod_{j=i+1}^{k}(1-p_j) = 1-(k-i)x$ and $\prod_{j=i}^{k}(1-p_j) = 1-(k-i+1)x$. Thus, if $i > \ell$, then $y_i = p_i = \frac{x}{1-(k-i)x}$ and

$$\begin{aligned} q_i = \prod_{j<i}(1-y_j) = (1-cx)^{\ell}\cdot\prod_{j=\ell+1}^{i-1}(1-p_i) &= (1-cx)^{\ell}\cdot\frac{\prod_{j=\ell+1}^{k}(1-p_i)}{\prod_{j=i}^{k}(1-p_i)} \\ &= (1-cx)^{\ell}\cdot\frac{1-(k-\ell)x}{1-(k-i+1)x}. \end{aligned}$$

Let $t = (k-\ell)x$ and $s = \ell x$ for notation simplicity. We have, for small x

$$\frac{q_i y_i}{x} = \begin{cases} \frac{1-e^{-cx}}{x}(1-cx)^{i-1} \approx c\cdot e^{-c\cdot(i-1)x}, & i \leqslant \ell \\ (1-cx)^{\ell}\cdot\frac{(1-t)}{(1-(k-i)x)\cdot(1-(k-i+1)x)} \approx e^{-c\cdot s}\frac{(1-t)}{(1-t+(i-\ell)x)^2}, & i > \ell \end{cases}$$

Consequently, we have that for small x

$$\begin{aligned} \sum_{i=1}^{k} x\cdot\left(1-e^{-\frac{q_i y_i}{x}}\right) &= \sum_{i=1}^{\ell} x\cdot\left(1-e^{-\frac{q_i y_i}{x}}\right) + \sum_{i=\ell+1}^{k} x\cdot\left(1-e^{-\frac{q_i y_i}{x}}\right) \\ &= \sum_{i=1}^{\ell} x\cdot\left(1-e^{-c\cdot e^{-c\cdot(i-1)x}}\right) + \sum_{i=\ell+1}^{k} x\cdot\left(1-e^{-e^{-c\cdot s}\frac{(1-t)}{(1-t+(i-\ell)x)^2}}\right) \\ &\geqslant \int_0^s 1-e^{-ce^{-cz}}\,\mathrm{d}z + \int_0^t 1-e^{-e^{-cs}\cdot\frac{1-t}{(1-t+z)^2}}\,\mathrm{d}z \end{aligned}$$

Furthermore, since $\frac{x}{1-t} \approx \frac{x}{1-(k-\ell-1)x} = p_{\ell+1} \leqslant 1-e^{-cx} \approx cx$, we have $t \leqslant 1-\frac{1}{c}$.

To finish the proof, it suffices to lower bound the following function

$$h_2(s,t) \stackrel{\text{def}}{=\!=} \left(\int_0^s 1 - e^{-ce^{-cz}} \mathrm{d}z + \int_0^t 1 - e^{-e^{-cs} \cdot \frac{1-t}{(1-t+z)^2}} \mathrm{d}z \right) / (s+t),$$

subject to $s \in [0,1], t \in \left[0, 1-\frac{1}{c}\right], s+t \in (0,1]$.

We use numerical methods to show $h_2(s,t) \geqslant h_2\left(\frac{1}{c}, 1-\frac{1}{c}\right) > 0.503$ when $c = 1.7$. The details are in the full version.

5 Improved Analysis: Regular Graphs

In this section, we prove a stronger performance guarantee of Greedy for regular graphs. According to the definition of log-normalized regular graph, each edge with log-normalized weight w_e can be substituted by a set of consecutive "small" edges with the same total log-normalized weight. For the ease of presentation, we assume that all edges are infinitesimal within this section. Let $x_e = \frac{w_e}{c}$ and $y_e = 1 - e^{-w_e}$ for all $e \in E$ as defined in Section 3.

Define the following two types of contributions of each edge $e \in E_{vu}$, where $u \in L$ and $v \in R$:

$$\mathrm{I}(e) \stackrel{\text{def}}{=\!=} x_e \cdot \left(1 - \exp\left(-\frac{q_u(e) \cdot y_e}{x_e}\right)\right);$$

and $\mathrm{II}(e) \stackrel{\text{def}}{=\!=} y_e \cdot \left(\mathbf{Pr}\left[\overline{M_u}(e)\right] - \mathbf{Pr}\left[Q_u(e)\right]\right)$.

In Section 3, we estimated performance of Greedy with a lower bound of $\sum_{e \in E} \mathrm{I}(e)$. Recall that this bound corresponds to the event that edge e is the first realized edge of u. It turns out that we can add an extra term $\mathrm{II}(e)$ on top of $\mathrm{I}(e)$ to have a better bound on the probability that edge e is matched. The term $\mathrm{II}(e)$ corresponds to the event that e is not the first realized edge of u, but u is still unmatched before e. Formally, we have the following Lemma 12, where coefficient $e^{-c-ce^{-c}}$ in front of $\mathrm{II}(e)$ ensures that the event from which we get extra gain is disjoint with the events from which we obtain the contribution of the first kind. I.e., we avoid double counting.

Within this section, we use computer assisted calculations in several places. We state all our lemmas in the case when $c = 2$ to highlight the improvement of our analysis over the competitive ratio of 0.532. We remark that our analysis generalizes for a wide range of the parameter c. The proofs of Lemmas 12 and 13 can be found in the full version.

▶ **Lemma 12.** *When $c = 2$,*

$$\mathsf{ALG} \geqslant \sum_{e \in E} \left(I(e) + e^{-c-ce^{-c}} \cdot II(e) \right). \tag{6}$$

By Lemma 4, the $\mathrm{I}(e)$ term alone is sufficient to show that Greedy is 0.532-competitive for 2-regular graphs. Next, we study the $\mathrm{II}(e)$ term.

Let $\delta_u = \mathbf{Pr}[\overline{M_u}] - \mathbf{Pr}[Q_u]$ for each vertex $u \in L$ at the end of algorithm's execution. Note that $\mathbf{Pr}[\overline{M_u}(e)] \geqslant \mathbf{Pr}[Q_u(e)]$ for all $e \in E_u$, because u cannot be matched at the moment of edge e arrival, if u had no realized edges. Thus $\delta_u \geqslant 0$. Similar to Lemma 3, we fix a vertex $u \in L$ and study the sum of $\mathrm{II}(e)$ for all edges $e \in E_u$.

▶ **Lemma 13.** *For any vertex $u \in L$, when $c = 2$,*

$$\sum_{e \in E_u} II(e) = \sum_{e \in E_u} y_e \cdot \left(\mathbf{Pr}\left[\overline{M_u}(e)\right] - \mathbf{Pr}\left[Q_u(e)\right]\right) \geqslant 1.98 \cdot \delta_u^2.$$

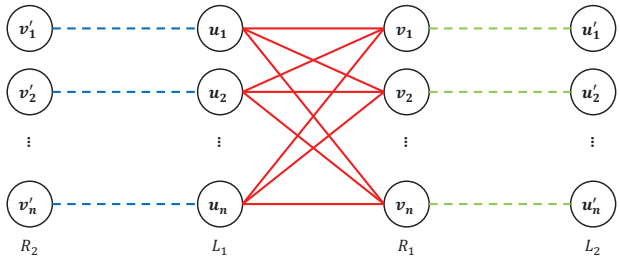

Figure 2 Hard instance for any algorithm.

▶ **Theorem 14.** *GREEDY is 0.552-competitive on 2-regular graphs.*

Proof. We have that

$$\mathsf{ALG} = \sum_{u\in L} \mathbf{Pr}\left[M_u\right] = \sum_{u\in L} \left(1 - \mathbf{Pr}\left[\overline{M_u}\right]\right) = \sum_{u\in L} \left(1 - e^{-c} - \delta_u\right).$$

We recall definition of $h_1(c) \stackrel{\text{def}}{=\!=} \int_0^1 \left(1 - e^{-ce^{-cz}}\right) \mathrm{d}z$ from Section 3. Then,

$$\begin{aligned}
&|L| \cdot (1 - e^{-c}) - \sum_{u\in L} \delta_u \\
\geqslant &\sum_{e\in E} \left(\mathrm{I}(e) + e^{-c-ce^{-c}} \cdot \mathrm{II}(e)\right) && \text{(by Lemma 12)} \\
\geqslant &\sum_{u\in L} \left(h_1(c) + e^{-c-ce^{-c}} \cdot 1.98 \cdot \delta_u^2\right) && \text{(by Lemma 4, 13)} \\
\geqslant &h_1(c) \cdot |L| + 1.98 \cdot e^{-c-ce^{-c}} \cdot \frac{\left(\sum_{u\in L} \delta_u\right)^2}{|L|}. && \text{(by Cauchy-Schwarz inequality)}
\end{aligned}$$

Let $\Delta = \frac{\sum_{u\in L} \delta_u}{|L|}$. We rearrange the above inequality and get the following for $c = 2$.

$$(1 - e^{-2} - h_1(2)) \geqslant \Delta + 1.98 \cdot e^{-2-2e^{-2}} \cdot \Delta^2.$$

Solving the inequality numerically, we have that $\Delta \leqslant 0.312$. Therefore,

$$\mathsf{ALG} = (1 - e^{-2} - \Delta) \cdot |L| \geqslant (1 - e^{-2} - 0.312) \cdot |L| \geqslant 0.552 \cdot |L|.$$ ◀

6 Problem Hardness

In this section, we present an upper bound of $\frac{2}{3} \approx 0.667$ for all online algorithms. Consider the graph shown in Figure 2. We use $L_1 = \{u_i\}_{i=1}^n, R_1 = \{v_j\}_{j=1}^n, L_2 = \{u'_i\}_{i=1}^n$ and $R_2 = \{v'_j\}_{j=1}^n$ to denote the vertices in the graph. The edges are defined as the following:

1. For each pair of $(u, v) \in L_1 \times R_1$, let there be an edge (u, v) with existence probability 1. We call them type-1 edges (red solid edges).
2. For each $i \in [n]$, let there be an edge (u_i, v'_i) with existence probability $\frac{1}{2}$. We call them type-2 edges (blue dashed edges).
3. For each $i \in [n]$, let there be an edge (u'_i, v_i) with existence probability $\frac{1}{2}$. We call them type-3 edges (green dashed edges).

Let the type-1 edges arrive first and then type-2 and type-3 edges.

▶ **Theorem 15.** *No algorithm is better than* $\frac{2}{3}$*-competitive.*

Proof. Note that there is no randomness for type-1 edges. If an algorithm matches k of them, there will be $\frac{n-k}{2}$ possible type-2 edges and $\frac{n-k}{2}$ type-3 edges in expectation. Thus any online algorithm matches no more than $k + \frac{n-k}{2} + \frac{n-k}{2} = n$ in expectation.

On the other hand, with high probability, there are at least $(0.5 - o(1)) \cdot n$ realized type-2 edges and at least $(0.5 - o(1)) \cdot n$ realized type-3 edges. In this case, the prophet can match $(0.5 - o(1)) \cdot n$ type-2 and type-3 edges respectively and then $0.5 \cdot n$ type-1 edges. In total, the prophet matches $(1.5 - o(1)) \cdot n$ edges with high probability. That is, $\mathsf{OPT} \geqslant (1.5 - o(1)) \cdot n$ when $n \to \infty$. ◀

References

1. Marek Adamczyk, Fabrizio Grandoni, and Joydeep Mukherjee. Improved approximation algorithms for stochastic matching. In *ESA*, volume 9294 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2015.
2. Itai Ashlagi, Maximilien Burq, Chinmoy Dutta, Patrick Jaillet, Amin Saberi, and Chris Sholley. Edge weighted online windowed matching. In *EC*, pages 729–742. ACM, 2019.
3. Bahman Bahmani and Michael Kapralov. Improved bounds for online stochastic matching. In *ESA (1)*, volume 6346 of *Lecture Notes in Computer Science*, pages 170–181. Springer, 2010.
4. Nikhil Bansal, Anupam Gupta, Jian Li, Julián Mestre, Viswanath Nagarajan, and Atri Rudra. When LP is the cure for your matching woes: Improved bounds for stochastic matchings. *Algorithmica*, 63(4):733–762, 2012.
5. Alok Baveja, Amit Chavan, Andrei Nikiforov, Aravind Srinivasan, and Pan Xu. Improved bounds in stochastic matching and optimization. *Algorithmica*, 80(11):3225–3252, 2018.
6. Benjamin Birnbaum and Claire Mathieu. On-line bipartite matching made simple. *ACM SIGACT News*, 39(1):80–87, 2008.
7. Niv Buchbinder, Kamal Jain, and Joseph Naor. Online primal-dual algorithms for maximizing ad-auctions revenue. In *ESA*, volume 4698 of *Lecture Notes in Computer Science*, pages 253–264. Springer, 2007.
8. Niv Buchbinder, Danny Segev, and Yevgeny Tkach. Online algorithms for maximum cardinality matching with edge arrivals. *Algorithmica*, 81(5):1781–1799, 2019.
9. Ning Chen, Nicole Immorlica, Anna R. Karlin, Mohammad Mahdian, and Atri Rudra. Approximating matches made in heaven. In *ICALP (1)*, volume 5555 of *Lecture Notes in Computer Science*, pages 266–278. Springer, 2009.
10. Ilan Reuven Cohen and David Wajc. Randomized online matching in regular graphs. In *SODA '18*, pages 960–979. SIAM, 2018.
11. Kevin P. Costello, Prasad Tetali, and Pushkar Tripathi. Stochastic matching with commitment. In *ICALP (1)*, volume 7391 of *Lecture Notes in Computer Science*, pages 822–833. Springer, 2012.
12. Nikhil R. Devanur and Kamal Jain. Online matching with concave returns. In *STOC*, pages 137–144. ACM, 2012.
13. Nikhil R. Devanur, Kamal Jain, and Robert D. Kleinberg. Randomized primal-dual analysis of RANKING for online bipartite matching. In *SODA*, pages 101–107. SIAM, 2013.
14. Leah Epstein, Asaf Levin, Danny Segev, and Oren Weimann. Improved bounds for online preemptive matching. In *STACS*, volume 20 of *LIPIcs*, pages 389–399. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2013.
15. Tomer Ezra, Michal Feldman, Nick Gravin, and Zhihao Gavin Tang. Online stochastic max-weight matching: Prophet inequality for vertex and edge arrival models. In *EC '20*, pages 769–787. ACM, 2020.
16. Jon Feldman, Aranyak Mehta, Vahab S. Mirrokni, and S. Muthukrishnan. Online stochastic matching: Beating $1 - 1/e$. In *FOCS*, pages 117–126. IEEE Computer Society, 2009.

17 Buddhima Gamlath, Sagar Kale, and Ola Svensson. Beating greedy for stochastic bipartite matching. In *SODA*, pages 2841–2854. SIAM, 2019.

18 Buddhima Gamlath, Michael Kapralov, Andreas Maggiori, Ola Svensson, and David Wajc. Online matching with general arrivals. In *FOCS*, pages 26–37. IEEE Computer Society, 2019.

19 Gagan Goel and Aranyak Mehta. Online budgeted matching in random input models with applications to adwords. In *SODA*, pages 982–991, 2008. URL: `http://dl.acm.org/citation.cfm?id=1347082.1347189`.

20 Nikolai Gravin and Hongao Wang. Prophet inequality for bipartite matching: Merits of being simple and non adaptive. In *EC*, pages 93–109. ACM, 2019.

21 Guru Prashanth Guruganesh and Sahil Singla. Online matroid intersection: Beating half for random arrival. In *IPCO*, volume 10328 of *Lecture Notes in Computer Science*, pages 241–253. Springer, 2017.

22 Zhiyi Huang, Ning Kang, Zhihao Gavin Tang, Xiaowei Wu, Yuhao Zhang, and Xue Zhu. How to match when all vertices arrive online. In *STOC*, pages 17–29. ACM, 2018.

23 Zhiyi Huang, Binghui Peng, Zhihao Gavin Tang, Runzhou Tao, Xiaowei Wu, and Yuhao Zhang. Tight competitive ratios of classic matching algorithms in the fully online model. In *SODA*, pages 2875–2886. SIAM, 2019.

24 Patrick Jaillet and Xin Lu. Online stochastic matching: New algorithms with better bounds. *Math. Oper. Res.*, 39(3):624–646, 2014.

25 Chinmay Karande, Aranyak Mehta, and Pushkar Tripathi. Online bipartite matching with unknown distributions. In *STOC*, pages 587–596, 2011.

26 Richard M. Karp, Umesh V. Vazirani, and Vijay V. Vazirani. An optimal algorithm for on-line bipartite matching. In *STOC*, pages 352–358, 1990.

27 Robert Kleinberg and S. Matthew Weinberg. Matroid prophet inequalities and applications to multi-dimensional mechanism design. *Games and Economic Behavior*, 113:97–115, 2019.

28 László Lovász and Michael D Plummer. *Matching theory*. Providence, R.I. : AMS Chelsea Pub, 2009. Originally published: Amsterdam; New York : North-Holland, 1986. URL: `https://ebookcentral.proquest.com/lib/qut/detail.action?docID=4832190`.

29 Mohammad Mahdian and Qiqi Yan. Online bipartite matching with random arrivals: an approach based on strongly factor-revealing LPs. In *STOC*, pages 597–606, 2011.

30 Vahideh H. Manshadi, Shayan Oveis Gharan, and Amin Saberi. Online stochastic matching: Online actions based on offline statistics. *Math. Oper. Res.*, 37(4):559–573, 2012.

31 Andrew McGregor. Finding graph matchings in data streams. In *APPROX-RANDOM*, volume 3624 of *Lecture Notes in Computer Science*, pages 170–181. Springer, 2005.

32 Aranyak Mehta. Online matching and ad allocation. *Foundations and Trends in Theoretical Computer Science*, 8(4):265–368, 2013.

33 Aranyak Mehta, Amin Saberi, Umesh V. Vazirani, and Vijay V. Vazirani. Adwords and generalized online matching. *J. ACM*, 54(5):22, 2007.

34 Joseph (Seffi) Naor and David Wajc. Near-optimum online ad allocation for targeted advertising. *ACM Trans. Economics and Comput.*, 6(3–4):16:1–16:20, 2018.

35 Ashwinkumar Badanidiyuru Varadaraja. Buyback problem – approximate matroid intersection with cancellation costs. In *ICALP (1)*, volume 6755 of *Lecture Notes in Computer Science*, pages 379–390. Springer, 2011.

36 Yajun Wang and Sam Chiu-wai Wong. Two-sided online bipartite matching and vertex cover: Beating the greedy algorithm. In *ICALP (1)*, volume 9134 of *Lecture Notes in Computer Science*, pages 1070–1081. Springer, 2015.