Max-Weight Online Stochastic Matching: Improved Approximations Against the Online Benchmark

Mark Braverman* *Princeton University*
Mahsa Derakhshan *UC Berkeley*
Antonio Molina Lovett *Princeton University*

In this paper, we study max-weight stochastic matchings on online bipartite graphs under both vertex and edge arrivals. We focus on designing polynomial time approximation algorithms with respect to the online benchmark, which was first considered by Papadimitriou, Pollner, Saberi, and Wajc [EC'21].

In the vertex arrival version of the problem, the goal is to find an approximate max-weight matching of a given bipartite graph when the vertices in one part of the graph arrive online in a fixed order with independent chances of failure. Whenever a vertex arrives we should decide, irrevocably, whether to match it with one of its unmatched neighbors or leave it unmatched forever. There has been a long line of work designing approximation algorithms for different variants of this problem with respect to the offline benchmark (prophet). Papadimitriou et al., however, propose the alternative *online* benchmark and show that considering this new benchmark allows them to improve the 0.5 approximation ratio, which is the best ratio achievable with respect to the offline benchmark. They provide a 0.51-approximation algorithm which was later improved to 0.526 by Saberi and Wajc [ICALP'21]. The main contribution of this paper is designing a simple algorithm with a significantly improved approximation ratio of $(1 - 1/e)$ for this problem.

We also consider the edge arrival version in which, instead of vertices, edges of the graph arrive in an online fashion with independent chances of failure. Designing approximation algorithms for this problem has also been studied extensively with the best approximation ratio being 0.337 with respect to the offline benchmark. This paper, however, is the first to consider the online benchmark for the edge arrival version of the problem. For this problem, we provide a simple algorithm with an approximation ratio of 0.5 with respect to the online benchmark.

*Research supported in part by the NSF Alan T. Waterman Award, Grant No. 1933331, a Packard Fellowship in Science and Engineering, and the Simons Collaboration on Algorithms and Geometry.

1 Introduction

The extensive literature on online Bayesian selection algorithms mainly focuses on the *competitive ratio*. That is, how well the algorithm performs against the optimal *offline* solution. Competing against such a strong benchmark often leads to pessimistic outcomes. For example, it is well-known that even for the single item version of the online Bayesian selection problem, the prophet inequality problem, no online algorithm can be better than $1/2$-competitive.

Another natural objective would be to compete with the best *online* solution. For many variants of online Bayesian selection problems (when the input is generated stochastically) one can write a dynamic program that makes the best decision at any point — hence the objective function is well defined. However, these algorithms are rarely computationally efficient. Indeed, Papadimitriou, Pollner, Saberi, and Wajc [23] show that a variant of the *online stochastic matching problem* is PSPACE-hard to approximate within some small constant. Thus, they initiate studying approximation algorithms for this problem with respect to the online benchmark. That is the solution found by an algorithm that has unlimited computational power, but is unaware of the part of the input that has not arrived.

In this paper, we study max-weight stochastic matchings on online bipartite graphs under both vertex and edge arrivals. Our main focus is on designing polynomial time approximation algorithms with respect to the online benchmark.

The Vertex arrival model. The goal in this problem is to find a large-weight matching of a bipartite graph when vertices in one part of the graph are online, arriving in a fixed order, each with an independent chance of failure. The vertices in the other part are present from the beginning thus referred to as the offline vertices. The graph, the arrival order of the online vertices, and their chances of failure are known from the beginning. The only unknown is whether a vertex actually arrives or if it fails. If a vertex does not arrive (i.e., fails), we do nothing about it. Otherwise, we either match it irrevocably to one of its unmatched neighbors or leave it unmatched forever. Papadimitriou et al. [23] refer to this problem as the RideHail problem due to its applications in ride hailing. However, it also models scenarios in other types of matching markets such as labor markets, online advertising, etc.

There has been a long line of work designing approximation algorithms for this problem (and its variants) with respect to the offline benchmark (see [19, 1, 15, 20, 11, 23] and the references within.) The best known algorithm with respect to this benchmark achieves a tight $1/2$ approximation ratio [12]. In their recent work, Papadimitriou et al. [23] show that this ratio can be improved to 0.51 if one considers the online benchmark instead. The online benchmark here is defined as a max-weight matching found by an algorithm that has unlimited computational power but does not know the arrival/failure of the future vertices. This approximation ratio was later improved to 0.526 using a machinery developed by Saberi and Wajc [24] for an online edge coloring problem. In this work, we design a simple algorithm with a significantly improved approximation ratio of $(1 - 1/e) \approx 0.632$ with respect to the online benchmark.

Result 1. (See Theorem 1) There exists a polynomial time algorithm for the online bipartite stochastic matching problem under (one-sided) vertex arrivals which finds a matching of weight at least a $(1 - 1/e)$ fraction of the one found by the best online algorithm.

Similar to Papadimitriou et al. [23], we also consider a more general version of the problem, which is also studied by [12, 11, 10], where upon arrival of a vertex, weights of its edges are drawn

from a known joint distribution. However, weights of edges incident to different online vertices are still independent. In Section 3.6 we explain how our algorithm and analysis can be extended to get the same approximation ratio of $(1 - 1/e)$ for this more general problem.

The Edge arrival model. The only difference between this problem and the vertex arrival version is that here, instead of the vertices, edges are online. Similarly, the goal in this problem is to find a large-weight matching of a bipartite graph when edges are online, arriving in a fixed order, each with an independent chance of failure. The graph, the arrival order of the edges and their chances of failure are known from the beginning. The only unknown is whether an edge actually arrives or if it fails. If an edge does not arrive (i.e., fails), we do nothing about it. Otherwise, we decide irrevocably whether to add it to our matching or not. See [16] for potential applications of this problem.

Designing approximation algorithms for the edge arrival version of the problem has also been studied extensively (see [6, 17, 11, 13] and the references within), with the best approximation ratio being 0.337 with respect to the offline benchmark [11]. It is also known that with respect to this benchmark it is not possible to achieve an approximation ratio better than $4/9$ [17]. This paper, however, is the first to consider the online benchmark for the edge arrival version of the problem. For this problem, we provide a simple algorithm with an approximation ratio of 0.5 with respect to the online benchmark.

Result 2. (See Theorem 2) There exists a polynomial time algorithm for the online bipartite stochastic matching problem under edge arrivals which finds a matching of weight at least a 1/2 fraction of the one found by the best online algorithm.

1.1 Our Techniques

For both vertex and edge arrival versions of the problem, we design LP-based algorithms consisting of an LP and a rounding procedure. The LP which we borrow from [25] outputs a fractional solution $\mathbf{x}$, where for any edge e, x_e can be interpreted as the probability of this edge joining the matching. Papadimitriou et al. [23] also use the same LP and furtur give a lower-bound of 0.875 for its integrality gap. Other than the basic matching constraints, the LP has an additional natural constraint which is crucial for separating the online and offline solutions. This constraint relies on the fact that whether a vertex/edge fails or not is independent of any decision made by the algorithm for the vertices/edges arriving before that. Thus, for instance, in the vertex arrival version, this constraint states that for any edge $e = (v, u)$, the offline vertex u should be unmatched with probability at least x_e/p_v before the arrival of vertex v. Here, p_v denotes the probability of v not failing (i.e. arriving). We explain this in more detail in Section 3.1. In this section, we focus on discussing our rounding procedure for the vertex arrival model to present the flavor of our work.

Our rounding procedure. To round the solution of the LP, we design a simple random rounding procedure. Upon arrival of a vertex v, it may receive matching proposals from its unmatched neighbors. If it receives any proposals, it accepts the best one (the one with the largest weight) and joins the matching. Otherwise, it remains unmatched. The process of sending proposals is as follows: When v arrives, any of its unmatched neighbors decides independently at random whether to send a proposal. The probability of sending these proposals is set in a way that for any edge,

the probability of it being proposed is lower-bounded by x_e. This is achievable in particular due to the LP constraint discussed above.

To be able to highlight the properties of our algorithm which allow us to improve the algorithm proposed by Papadimitriou et al. [23], we will give a brief overview of their algorithm below.

Algorithm proposed by Papadimitriou et al. Let us emphasize that this is just a brief and paraphrased overview of the algorithm proposed in [23] which we include for the sake of comparison. They start with the same LP that we use. However, their rounding procedure is different. Upon arrival of an online vertex v_t, it picks one of its neighbors randomly proportional to the probabilities given by the LP and sends a proposal to it. If the neighbor is unmatched it accepts the proposal with some probability and matches with v_t. These probabilities are set in a way that each edge joins the matching with probability at least $0.51x_e$. However, some vertices are not able to satisfy this for their edges by sending only a single proposal. The algorithm gives such vertices a second chance to send another proposal if their first proposal is not accepted, and this allows them to guarantee a matching probability of $0.51x_e$ for all the edges. In the rest of the paper, we will refer to this algorithm as PPSW (the authors' initials.)

As the first difference, in PPSW the offline vertices receive the proposals and need to decide whether to accept a proposal without knowing their future proposals. In our algorithm however, since the online vertices are the ones receiving the proposals they can make a decision while knowing all their options. This is particularly helpful since edges are weighted and an online vertex has the option of picking the one with the highest weight. With this advantage, however, comes a new challenge. We cannot guarantee that all the edges will join the matching with a large probability. Indeed, some low-weight edges may have a very small probability. To overcome this, instead of analyzing the rounding loss for any edge, we lower-bound the loss imposed on any online vertex due to the rounding procedure.

The second difference is in the number of proposals a vertex can send. We do not limit the number of proposals a vertex can send. Indeed, a vertex can send proposals as long as it is unmatched. PPSW on the other hand, imposes the limit of two proposal per any online vertex. This is due to the way their analysis works. The main meat of their analysis is upper-bounding the positive correlation between the matching status of the offline vertices, and allowing the vertices to send many proposals worsens the correlation. Let us first explain why absence of positive correlation is desirable. A key part of our analysis is proving that our algorithm satisfies the following property: whenever an online vertex v arrives, the probability of it *not* receiving any proposals from any subset S of its neighbors is at most

$$\prod_{u \in S} (1 - x_{(v,u)}/p_v). \tag{1}$$

As mentioned above, the probability of v receiving a proposal from any of its neighbors u is at least $x_{(v,u)}$. Since v arrives with probability p_v, the probability of it *not* receiving a proposal from a neighbor u is at most $(1 - x_{(v,u)}/p_v)$. Thus, property (1) follows directly if we were allowed to assume that the matching status of the vertices in S are independent when v arrives. It is not complicated to show that the same holds if they are not positively correlated. Unfortunately though, trying to prove this key property through positive correlation fails as we show via an example (See Section 3.5.) that these events can indeed be negatively correlated. However, we are still able to prove the existence of this property via a different method without concerning ourselves with the correlation between the matching status of the offline vertices. We even show that our analysis is

tight for an instance of the problem. Interestingly, in this instance our algorithm does not cause any positive correlation between the matching status of the offline vertices. (See Section 3.4.) This all means that positive correlation by itself is not the enemy. It only hurts us if it decreases the probability of a vertex receiving at least one proposal in comparison to the case of its neighbors being independent. Our approach is to carefully lower-bound the probability of this event using simple mathematical tools. See Lemma 3.4 for more details.

1.2 Further Related work

As we mentioned before, most of the literature on online Bayesian selection focuses on designing algorithms with respect to the offline benchmark which is often referred to as the prophet. It would be impossible to do justice to this extensive literature in this amount of space, thus we just briefly outline some of the most relevant works here. The study of prophet inequality problem, the single item version of the online Bayesian selection problem, was initiated by Krengel and Sucheston [21] who give an algorithm with competitive ratio of 1/2. Their seminal work was a starting point for studying more general versions of online Bayesian selection problems. The ones most related to multi-item prophet inequalities under matroid constraints [20], stochastic matching under vertex arrivals [19, 1, 15, 20, 11, 23] and stochastic matching under edge arrivals [6, 17, 11, 13].

It is worth mentioning that the connection between prophet inequalities and algorithmic mechanism design first discovered by Hajiaghayi, Kleinberg and Sandholm [18], has played a significant role in motivating the study of approximation algorithms for these online Bayesian selection problems. For more detailed related work on the topic of prophet inequality and also its relations to algorithmic mechanism design see [22, 8].

Max-weight matching on stochastic graphs has also been studied extensively under the query model. That is, similar to our model, there is an underlying stochastic graph, however, to know whether an edge exists it should be queried. Some works focus on having a small number of queries [26, 5, 4, 3] while others require any queried and realized edge to join the matching [7, 9, 2, 14]. The main application of these models is for environments with costly queries such as organ exchange markets.

Paper organization. The rest of the paper is organized as follows. In Section 2, we provide a formal definition of our problems and some notions that we will use throughout the paper. Section 3 is about the vertex arrival model. In this section, we first present the algorithm and its analysis in 3.1 and 3.3 respectively. Later, in 3.4, we provide an example for which our algorithm causes positive correlation between the matching status of the offline vertices, and in 3.5 we show that the analysis of our algorithm is tight. Further, in 3.6 we explain how our results can be extended to a more general version of the problem where the weights of the edges connected to any online vertex are drawn from a joint distribution. Finally, we discuss the edge arrival version of the problem in Section 4 with the algorithm and its analysis being in 4.1 and 4.3 respectively.

2 Preliminaries

We are given a bipartite graph $G = (A, B, E)$ and a weight w_e for each edge $e \in E$. In the vertex arrival model, we also have a probability p_v for each $v \in A$ and a fixed order $(v_1, \dots, v_{|A|})$ over the vertices in A. Vertices in B are initially present, but vertices in A arrive online. At any time t, with probability p_{v_t} vertex v_t arrives (or is realized). If it does, we are allowed to match v_t irrevocably to one of its unmatched neighbors or else commit to leaving it unmatched forever. If it does not

arrive, we do nothing at time t. In the edge arrival model, similarly, we are also given a probability p_e for each $e \in E$ and a fixed order $(e_1, \ldots, e_{|E|})$ over the edges. At any time t, with probability p_{e_t} edge e_t arrives (or is realized). If it does, we should decide irrevocably whether to add it to our matching or lose it forever. Under both arrival models, the goal is to maximize the total weight of the edges we add to the matching.

In this paper, our focus is on designing a polynomial time algorithms for the above problem under both arrival models. We say that an algorithm $\mathcal{A}$ is an α-approximation if for any instance $\mathcal{I}$ of the problem, it satisfies

$$\mathbb{E}[\mathcal{A}(\mathcal{I})] \geq \alpha \mathbb{E}[\mathrm{OPT}_{\mathrm{on}}(\mathcal{I})],$$

where $\mathrm{OPT}_{\mathrm{on}}$ is the optimal online algorithm. That is an algorithm that has unlimited computational power, but its knowledge about the arrival of future vertices/edges is the same as ours.

For ease of notation, for any pair of edges e_1, e_2, we say $e_1 < e_2$ if e_1 arrives before e_2. We also use $e_1 < t$ to mean e_1 arrives before time t. (Note that in the vertex arrival model, an edge arrives whenever its online end-point arrives.) Also, when it is clear from context, we will use t to refer to the vertex v_t (or edge e_t), arriving at time t. Finally, we write our edges as ordered pairs, meaning that for any edge $(v, u) \in E$ we always have $v \in A$ and $u \in B$.

3 Vertex Arrivals

3.1 The Algorithm

We begin by writing a linear program that attempts to model the optimal (omnipotent) online algorithm's behavior. This LP is also used by Papadimitriou et al., however, for the sake of self-containment we explain it in detail here. For any edge $e \in E$, we have a variable x_e which represents the probability of e joining the matching in $\mathrm{OPT}_{\mathrm{on}}$. Here, the randomness can be over both the stochastic arrivals of the vertices and any random decisions made by the algorithm. We claim then that such x_e are feasible for the following LP:

$$\max_{\mathbf{x}} \quad \sum_{e \in E} w_e x_e \,, \tag{2}$$

$$\text{s.t.} \quad \sum_{e \ni v} x_e \leq p_v \qquad \forall\, v \in A \,, \tag{3}$$

$$\sum_{e \ni u} x_e \leq 1 \qquad \forall\, u \in B \,, \tag{4}$$

$$p_v \cdot (1 - \sum_{e' \ni u, e' < e} x_{e'}) \geq x_e \qquad \forall\, e = (v, u) \in E \,, \tag{5}$$

$$x_e \geq 0 \qquad \forall e \in E \,. \tag{6}$$

The first two constraints (4 and 3) are standard matching constraints since each vertex can be incident to at most one edge in the matching, and each $v \in A$ is unmatched with probability at least $1 - p_v$ (when it fails). Constraint 5 is however special to the online solution. It asserts that for any edge (v_t, u) the probability of u being unmatched before time t (the left-hand side) is at least x_e / p_{v_t}. This is due to the fact that arrival of vertex $v_t \in A$ is independent of whether u is matched before time t. (All vertices arrive independently, and a non-omniscient algorithm must have made all matching decisions independently of future arrivals.) If this constraint is not satisfied then u is unmatched with probability less that x_e / p_{v_t}. In this case, the probability that v_t arrives and u is still unmatched by time t is less than x_e, contradicting the definition of x_e.

Observation 3.1 ([25]). *Let* $\mathbf{x}$ *be an optimal solution of the LP. We have* $\mathrm{OPT}_{LP} \geq \mathrm{OPT}_{on}$ *where* $\mathrm{OPT}_{LP} = \sum_{e \in E} w_e x_e$.

3.2 The Rounding Procedure

The next step of the algorithm is rounding the fractional solution of the LP. For that, we design a simple rounding procedure that given any optimal solution of the LP (which is a fractional matching), outputs an integral matching. Later we will prove that the output of this algorithm is a $(1 - 1/e)$-approximate solution.

Our rounding procedure is very simple and natural. Whenever a vertex $v_t \in A$ arrives, we construct a random subset P of its unmatched neighbors as potential matches. Any unmatched neighbors decide independently at random whether to send a matching proposal to v_t and join subset P. If v_t receives at least one proposal (i.e., if P is nonempty), it accepts the best one (the one with the largest weight) and joins the matching. Otherwise, it remains unmatched forever. In our algorithm, we set the probability of sending proposals in a way that for any edge $e = (v_t, u) \in E$, it results in

$$\Pr[v_t \text{ receives a proposal from } u] \geq x_e.$$

This is achievable thanks to the constraint 5 of the LP which separates the online and offline benchmarks. In other words, it is not possible to satisfy this inequality for any arbitrary fractional matching, and this is where we use the fact that we are competing with the best online algorithm. To be able to satisfy this property, whenever v_t arrives and u is unmatched we need u to send a proposal to v_t with probability at least

$$\frac{x_e}{p_t \Pr[u \text{ is unmactched}]}.$$

This is of course achievable only if this number is not larger than one, which will be shown as a consequence of our analysis.

Algorithm 1. Rounding Procedure

Let $\mathbf{x}$ be an optimal solution of the LP.
Let $M \leftarrow \emptyset$ be a matching of E.
for $t \in |A|$ **do**
  $v \leftarrow v_t$
  Let set N_v denote neighbors of vertex v in graph G.
  $P \leftarrow \emptyset$.
  For any vertex $u \in N_v$, define $\alpha_u = \sum_{e \ni u, e < (v,u)} x_e$.
  For any vertex $u \in N_v$, if u is matched in M and $x_{(v,u)} > 0$, then with probability $\frac{x_{(v,u)}}{p_v(1-\alpha_u)}$ add edge (v, u) to set P independently.
  if P is non-empty and v_t is realized **then**
    Add edge $\arg\max_{e \in P} w_e$ to matching M.
  end if
end for
Return matching M.

3.3 The Analysis

The purpose of this section is proving that Algorithm 1 finds a $(1-1/e)$-approximate matching. Before proceeding with our analysis, we need to define some notations. In the rest of the paper, we use $\mathcal{WM}$ to represent the weighted matching outputted by Algorithm 1. Moreover, for any vertex v, if it is matched in $\mathcal{WM}$, (i.e., $v \in \mathcal{WM}$), we use $\mathcal{WM}(v)$ to represent the weight of its matching edge in $\mathcal{WM}$. Note that $\mathcal{WM}$ and $\mathcal{WM}(v)$ are both random variables. For any vertex $u \in B$ and any time t, we define

$$\alpha_{t,u} := \sum_{e \ni u, e < (v_t,u)} x_e.$$

Finally, for a given subset of vertices $S \subset B$, we define E_t^S to be the event in which all the vertices from S are matched before time t, and F_t^S to similarly be the event that all vertices in S are still free (unmatched) just before time t.

In our calculations, we will make use of the following lemma. However, to preserve the flow of the paper, we defer its proof to Section 5.

Lemma 3.2. *Let $S \subset B$ be a set of vertices. Suppose we associate each vertex $u \in S$ with a number $w_u \in \mathbb{R}$. Then*

$$\sum_{X \subset S} \Pr\left[E_t^{S \setminus X} \cap F_t^X\right] \prod_{u \in X} w_u = \sum_{X \subset S} \Pr\left[E_t^{S \setminus X}\right] \left(\prod_{u \in X} w_u\right) \prod_{u \in S \setminus X} (1 - w_u).$$

We can now begin our analysis with a crucial property of our algorithm. That is upper-bounding the probability of all the vertices in S being matched before time t for any $S \subset B$.

Lemma 3.3. *At any time t, for any subset of vertices $S \subset B$, we have*

$$\Pr[E_t^S] \leq \prod_{u \in S} \alpha_{t,u}. \tag{7}$$

Proof. We use proof by induction on t. Our claim holds for the base case of $t = 1$ as for $t = 1$, both sides of Equation 7 equal to zero for nonempty S (and one for $S = \emptyset$). Assuming that for some $t \geq 1$, this equation holds, we will prove it for $t + 1$. In other words, we will prove

$$\Pr[E_{t+1}^S] \leq \prod_{u \in S} (\alpha_{t,u} + x_{(t,u)}), \tag{8}$$

for any $S \subset B$. For now, we assume that

$$\alpha_{t+1,u} = \alpha_{t,u} + x_{(v_t,u)} < 1 \tag{9}$$

holds for all $u \in S$. For such u, we may define

$$\beta_{t,u} = \frac{x_{(v_t,u)}}{1 - \alpha_{t,u}} \in [0, 1). \tag{10}$$

We aim to show that S satisfies Equation 8. For E_{t+1}^S to occur, either all of S was matched already before time t, or some vertex $u \in S$ was matched exactly at time t, with the others matched before. This lets us compute

$$\Pr[E_{t+1}^S] = \Pr[E_t^S] + \sum_{u \in S} \Pr\left[E_t^{S \setminus \{u\}} \cap F_t^{\{u\}}\right] \Pr\left[(v_t, u) \in M \mid E_t^{S \setminus \{u\}} \cap F_t^{\{u\}}\right]$$

$$\leq \Pr[E_t^S] + \sum_{u \in S} \Pr\left[E_t^{S\setminus\{u\}} \cap F_t^{\{u\}}\right] \frac{x_{(v_t,u)}}{p_v(1-\alpha_{t,u})} p_v \tag{11}$$

(to be matched, (v_t, u) must have been added to P and v_t must have arrived, independently)

$$= \sum_{\substack{X \subset S \\ |X| \leq 1}} \Pr\left[E_t^{S\setminus X} \cap F_t^X\right] \prod_{u \in X} \beta_{t,u}$$

$$\leq \sum_{X \subset S} \Pr\left[E_t^{S\setminus X} \cap F_t^X\right] \prod_{u \in X} \beta_{t,u} \qquad \text{(Equation 10)}$$

$$= \sum_{Y \subset S} \Pr\left[E_t^{S\setminus Y}\right] \left(\prod_{u \in Y} \beta_{t,u}\right) \prod_{u \in S\setminus Y} (1-\beta_{t,u}) \qquad \text{(Lemma 3.2)}$$

$$\leq \sum_{Y \subset S} \left(\prod_{u \in S\setminus Y} \alpha_{t,u}\right) \left(\prod_{u \in Y} \beta_{t,u}\right) \prod_{u \in S\setminus Y} (1-\beta_{t,u})$$

(inductive hypothesis, and Equation 10)

$$= \prod_{u \in S} (\beta_{t,u} + \alpha_{t,u}(1-\beta_{t,u}))$$

$$= \prod_{u \in S} \frac{x_{(v_t,u)} + \alpha_{t,u}(1-\alpha_{t,u}-x_{(v_t,u)})}{1-\alpha_{t,u}}$$

$$= \prod_{u \in S} (x_{(v_t,u)} + \alpha_{t,u}),$$

which is exactly Equation 8.

Before we complete our proof, we must still consider $S \subset B$ where, for at least some $u \in S$, the inequality from Equation 9 is violated. Let $S' \subset S$ denote the vertices satisfying Equation 9. Then

$$\Pr[E_{t+1}^S] \leq \Pr[E_{t+1}^{S'}] \leq \prod_{u \in S'} \alpha_{t+1,u} \leq \prod_{u \in S} \alpha_{t+1,u},$$

since for any $u \in S \setminus S'$, we have $\alpha_{t+1,u} > 1$ (and all other $\alpha_{t,u}$ are non-negative). □

Lemma 3.4. *In Algorithm 1, at any time t, for any subset of vertices $S \subset B$, the probability that none of the vertices in S joins P is upper-bounded by*

$$\prod_{u \in S} (1 - x_{(t,u)}/p_t).$$

Proof. Pick some $S \subset B$. In order for none of the vertices from S to join P, then each $u \in S$ that is still unmatched must fail to be sampled with probability $\frac{x_{(v_t,u)}}{p_t(1-\alpha_u)} = \beta_{t,u}/p_t$, using $\beta_{t,u}$ as defined in the proof of Lemma 3.3. We can thus bound this probability to be at most

$$\sum_{X \subset S} \Pr\left[E_t^{S\setminus X} \cap F_t^X\right] \prod_{u \in X} (1 - \beta_{t,u}/p_t)$$

$$= \sum_{Y \subset S} \Pr\left[E_t^{S\setminus Y}\right] \left(\prod_{u \in Y} (1-\beta_{t,u}/p_t)\right) \prod_{u \in S\setminus Y} \beta_{t,u}/p_t \qquad \text{(Lemma 3.2)}$$

$$\leq \sum_{Y \subset S} \left(\prod_{u \in S\setminus Y} \alpha_{t,u}\right) \left(\prod_{u \in Y} (1-\beta_{t,u}/p_t)\right) \prod_{u \in S\setminus Y} \beta_{t,u}/p_t \qquad \text{(Lemma 3.3)}$$

$$= \prod_{u \in S} (\alpha_{t,u}\beta_{t,u}/p_t + 1 - \beta_{t,u}/p_t)$$
$$= \prod_{u \in S} \frac{\alpha_{t,u}x_{(t,u)}/p_t + (1-\alpha_{t,u}) - x_{(t,u)}/p_t}{1-\alpha_{t,u}}$$
$$= \prod_{u \in S} (1 - x_{(t,u)}/p_t). \qquad \square$$

Lemma 3.5. *For any vertex $v_t \in A$, and any non-negative number w we have:*

$$\Pr[\mathcal{WM}(v_t) \geq w] \geq (1-1/e) \sum_{e \ni v_t, w_e \geq w} x_e. \tag{12}$$

Proof. Fix some $v_t \in A$ and $w \geq 0$. Let $S \subset B$ denote the set of all $u \in B$ such that $w_{(v_t,u)} \geq w$. Then as long as some vertex from S is added to P at time t, and vertex v_t arrives, then $\mathcal{WM}(v_t) \geq w$ will hold. Note that the arrival of v_t is independent of P, so we can compute

$$\begin{aligned}
\Pr[\mathcal{WM}(v_t) \geq w] &= p_t \Pr[S \cap P \neq \emptyset] \\
&= p_t - p_t \prod_{u \in S} (1 - x_{(t,u)}/p_t) && \text{(Lemma 3.4)} \\
&\geq p_t - p_t \left(1 - \sum_{u \in S} \frac{x_{(t,u)}}{p_t |S|}\right)^{|S|} && \text{(AM-GM)} \\
&\geq p_t - p_t \exp\left[-\sum_{u \in S} x_{(t,u)}/p_t\right] \\
&\geq p_t - p_t \left(1 - (1-1/e) \sum_{u \in S} x_{(t,u)}/p_t\right) && \text{(convexity)} \\
&= (1-1/e) \sum_{u \in S} x_{(t,u)}. && \square
\end{aligned}$$

Theorem 1. *Algorithm 1 outputs a $(1-1/e)$-approximate matching, that is*

$$\sum_{v_t \in A} \mathbb{E}[\mathcal{WM}(v_t)] \geq (1-1/e) \cdot \mathrm{OPT}_{on}.$$

Proof. By Observation 3.1 we know that $\mathrm{OPT}_{\mathrm{LP}}$ gives us an upper-bound for $\mathrm{OPT}_{\mathrm{on}}$, that is:

$$\sum_{e \in E} w_e x_e = \sum_{v_t \in A} \sum_{e \ni v_t} w_e x_e \geq \mathrm{OPT}_{\mathrm{on}}.$$

As a result, to prove this theorem, it suffices to show that for any vertex $v_t \in A$ we have

$$\mathbb{E}[\mathcal{WM}(v_t)] \geq (1-1/e) \sum_{e \ni v_t} w_e x_e,$$

which is the same as proving

$$\sum_{e \ni v_t} w_e x_e - \mathbb{E}[\mathcal{WM}(v_t)] \leq \left(\sum_{e \ni v_t} w_e x_e\right)/e. \tag{13}$$

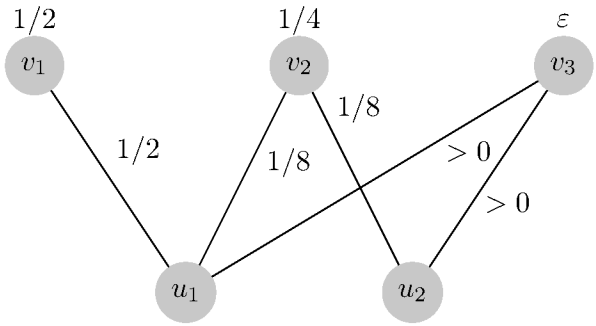
By definition, for any vertex $v_t \in A$ we can write the left-hand side of this inequality as

$$\begin{aligned}
\sum_{e \ni v_t} w_e x_e - \mathbb{E}[\mathcal{WM}(v_t)] &= \int_{w=0}^{\infty} \left(\sum_{e \ni v_t, w_e > w} x_e \right) - \int_{w=0}^{\infty} \Pr[\mathcal{WM}(v_t) > w] \\
&= \int_{w=0}^{\infty} \left(\sum_{e \ni v_t, w_e > w} x_e - \Pr[\mathcal{WM}(v_t) > w] \right) \\
&\leq \int_{w=0}^{\infty} \left(\sum_{e \ni v_t, w_e > w} x_e \right) / e \qquad \text{(Lemma 3.5)} \\
&= \left(\sum_{e \ni v_t} w_e x_e \right) / e.
\end{aligned}$$

This proves Equation 13 and concludes the proof of the theorem. □

3.4 Positive Correlation

Much of the detail needed in our proof of Theorem 1 and related lemmas is due to handling potential correlation between the matched/unmatched status of the vertices in B. In particular, the proof of our main lemma (Lemma 3.4) could proceed fairly directly if we were allowed to assume that events $E_t^{\{u_1\}}, \ldots, E_T^{\{u_{|B|}\}}$ (the events that vertices in $u \in B$ are matched before any time t), are independent from each other. Similarly, if we had the notion of negative dependence used in [23], namely *negative association* of the indicator variables for the events $E_t^{\{u\}}$, this would also suffice to arrive at Lemma 3.4. In this section, we will show that a more involved analysis such as ours is in fact necessary since our algorithm sometimes causes positive correlation between these events. We construct a bipartite graph $G = (A, B, E)$ with $A = \{v_1, v_2, v_3\}$ and $B = \{u_1, u_2\}$, such that before time $t = 3$ the events $E_3^{\{u_1\}}$ and $E_3^{\{u_2\}}$ are in fact positively correlated. The edge set, along with the values of p_v and x_e for $v \in A$ and $e \in E$ are given in the following diagram:



One can easily verify that our solution x satisfies the LP constraints, and is optimal for certain values of w (in particular, when $w_{(v_1,u_1)} = 100$, and when $w_{(v_2,u_1)} = 2$ and $w_{(v_2,u_2)} = 1$, and small weights incident to v_3).

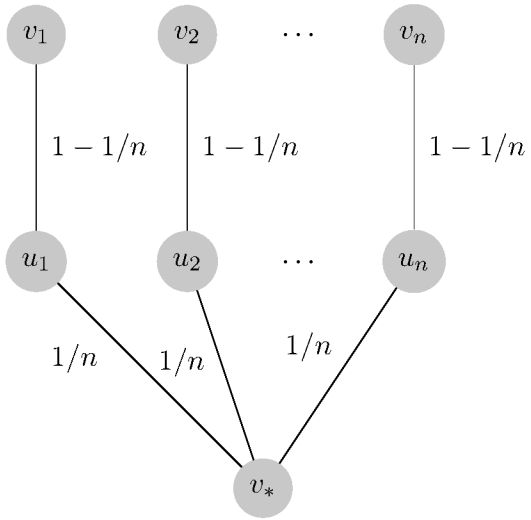
With probability $1/2$, the first vertex v_1 arrives, and is matched with u_1 with probability $\frac{1/2}{1/2} = 1$. Assuming this occurs, v_2 matches u_2 if it arrives (with probability $1/4$) and u_2 is added to P in the second step (with probability $\frac{1/8}{1/4} = 1/2$).

The other $1/2$ of the time, the first vertex v_1 does not arrive, so u_1 is added to P in the second step with probability $\frac{1/8}{(1/4)(1-1/2)} = 1$ when v_2 arrives with probability $1/4$, and no edges are matched otherwise. Overall, after time $t = 2$, both u_1 and u_2 are matched with probability $1/8$, neither is matched with probability $3/8$, and just u_1 is matched with probability $3/8 + 1/8 = 1/2$. The indicator variables for the events $E_3^{\{u_1\}}$ and $E_3^{\{u_2\}}$ will thus have positive covariance

$$\Pr\left[E_3^{\{u_1,u_2\}}\right] - \Pr\left[E_3^{\{u_1\}}\right]\Pr\left[E_3^{\{u_2\}}\right] = 1/8 - (5/8)(1/8) = 3/64 > 0.$$

3.5 Tightness of the Analysis

First, we show that our algorithm indeed loses the factor of $(1 - 1/e)$ compared to OPT_{on}. We construct the graph $G = (A, B, E)$, where $A = \{v_1, \ldots, v_n, v_*\}$ and $B = \{u_1, \ldots, u_n\}$. For each i, there are edges (v_i, u_i) and (v_*, u_i) with weights $1/n^2$ and 1 respectively. The vertices from A arrive in order $v_1, \ldots, v_n, v_*$, and we have $p_{v_i} = 1 - 1/n$ for all i and $p_{v_*} = 1$. Then, the unique optimal solution x to our LP is given in the following diagram (namely, $x_{(v_i,u_i)} = 1 - 1/n$ and $x_{(v_*,u_i)} = 1/n$).



Consider what our algorithm would do faced with this graph. For each i, it would add u_i to P with probability 1, then add (v_i, u_i) to our matching if v_i is realized. Hence, the probability that u_i is unmatched by the time we get to vertex v_* is exactly $1/n$, and is independent of all other vertices from B. The probability that v_* will have no neighbors unmatched is thus $(1 - 1/n)^n$.

We can now bound our algorithms expected matching weight to be at most

$$(1 - (1 - 1/n)^n) + n(1 - 1/n)(1/n^2),$$

which for large n gets arbitrarily close to $1-1/e$. On the other hand, a trivial online algorithm could instead never match any of the edges (v_i, u_i), and always take one of the edges (v_*, u_i), obtaining a matching with weight 1 always. The optimal online algorithm, OPT_{on}, would thus need to attain at least 1 in expectation, proving that our algorithm can never be $(1 - 1/e + \varepsilon)$-competitive for any $\varepsilon > 0$.

Note that in this example, for any vertex A at the time of its arrival, the matching status of its neighbors are independent. This intuitively means that the loss our algorithm incurs is not due to the correlation it causes between the matching status of the vertices.

3.6 Generalization of the Algorithm and Analysis

With our analysis complete, we can now extend our algorithm to a more general version of the vertex arrival model, allowing for distributions over edge weights. We will first describe the new model, which Papadimitriou, Pollner, Saberi, and Wajc also considered for their algorithm [23]. We will then explain the main considerations for adapting our algorithm and analysis from Sections 3.1–3.3 for this harder case.

General Vertex Arrival Model Just as in the original vertex arrival model (described in Section 2), we have a known bipartite graph $G = (A, B, E)$ and fixed order $(v_1, \ldots, v_{|A|})$ over the vertices in A. Vertices in B are initially present, but vertices in A arrive online in this order. However, rather than having fixed weights for all edges, with each vertex $v_t \in A$ arriving with a probability p_{v_t}, we instead realize a sample w^t from a distribution over possible weights for all edges incident to v_t.

Formally, for each time $1 \leq t \leq |A|$, there is a joint distribution $\mathcal{D}_t$ with finite support over non-negative assignments of weights for all edges incident to v_t. At time t, we draw a sample $w^t \sim \mathcal{D}_t$. This tells us the realized weight w^t_e for each edge $e = (v_t, u)$ incident to v_t. As before, we may now choose to match v_t irrevocably to one of its unmatched neighbours. The goal is to maximize the total realized weight of all edges in our matching, given by

$$\sum_{(v_t,u)\in M} w^t_{(v_t,u)},$$

where M denotes the set of edges in our final matching.

We note that this is indeed a generalization of our original vertex arrival model, which can be represented here by $\mathcal{D}_t$ yielding the vector of values $(w_{(v_t,u)})_u$ with probability p_t, and the zero vector with probability $1 - p_t$.

Modified Algorithm To begin, we modify our LP from Section 3.1 to yield another LP relaxation under this more general model, using the same natural extension as given in [23].

Since our distributions $\mathcal{D}_t$ are assumed to be finite, for each t, we let $p_{t,i}$ denote the probability mass for the i-th possibility (and i varies from 1 to the size of the support of $\mathcal{D}_t$), and define $w_{t,i,u}$ to be the weight assigned to edge (v_t, u) in this case. We will have variables $y_{t,i,u}$ representing the probability of v_t being matched to u with value $w_{t,i,u}$ in OPT_{on}. These take the place of x_e from before (representing the probability of e being in our matching), so we can now give our modified LP:

$$\max_{\mathbf{y}} \quad \sum_{(v_t,u)\in E,i} w_{t,i,u} \cdot y_{t,i,u}\,, \tag{14}$$

$$\text{s.t.} \quad \sum_{(v_t,u)\in E} y_{t,i,u} \leq p_{t,i} \qquad \forall\, v_t \in A, i\,, \tag{15}$$

$$\sum_{t,i} y_{t,i,u} \leq 1 \qquad \forall\, u \in B\,, \tag{16}$$

$$p_{t,i} \cdot \Big(1 - \sum_{t'<t,i'} y_{t',i',u}\Big) \geq y_{t,i,u} \qquad \forall\, v_t \in A, u \in B, i\,, \tag{17}$$

$$y_{t,i,u} \geq 0 \qquad \forall v_t \in A, u \in B, i\,. \tag{18}$$

The actual rounding procedure of Algorithm 1 also needs modification. At each iteration of the loop, we will first sample $w^t \sim \mathcal{D}_t$, obtaining some possibility $\hat{i}$. We can define

$$\alpha_u = \sum_{t'<t,i} y_{t',i,u} \tag{19}$$

instead at Line 7. We will now add each vertex $u \in N_v$ to P independently with probability $\frac{y_{t,\hat{i},u}}{p_{t,\hat{i}}(1-\alpha_u)}$ instead of $\frac{x_{(v,u)}}{p_v(1-\alpha_u)}$ at Line 8. This is again easily seen to be well-defined, by Equation 17 from our modified LP. The maximum weight sampled neighbour will then be chosen based on the sampled weights $w_{t,\hat{i},u}$.

Modified Analysis Our analysis remains largely valid, and applies to this more general model with minor modifications.

To start, $\alpha_{t,u}$ must be defined as in Equation 19. This allows the proof of Lemma 3.3 to go through as written, replacing occurrences of $x_{(v_t,u)}$ with $\sum_i y_{t,i,u}$. We must also more carefully expand the probability at Equation 11, observing that

$$\Pr\left[(v_t,u) \in M \mid E_t^{S\setminus\{u\}} \cap F_t^{\{u\}}\right] \leq \sum_i p_{t,i} \frac{y_{t,i,u}}{p_{t,i}(1-\alpha_{t,u})} = \beta_{t,u}.$$

Lemma 3.4 needs slight adjustment to its statement. We can instead show that, for any i, *if we assume* that the i-th possible weight vector is drawn from $\mathcal{D}^t$, so the realized weight of (v_t,u) is $w_{t,i,u}$ for all u, *then* the probability that no vertex from S joins P is upper-bounded by

$$\prod_{u\in S}(1-y_{t,i,u}/p_{t,i}).$$

The proof now still holds, replacing all occurrences of $x_{(t,u)}$ with $y_{t,i,u}$, and replacing occurrences of p_t with $p_{t,i}$.

Lemma 3.5 can be modified similarly, again conditioning on the realization of $\mathcal{D}^t$, and making the same substitutions, defining $S \subset B$ as all $u \in B$ where $w_{t,i,u} \geq w$ for the assumed realization i. Finally, to extend Theorem 1 to this more general model, by taking expectations over the drawing of $w^t \sim \mathcal{D}^t$, it suffices to show for any time t and realization i that

$$\mathbb{E}[\mathcal{WM}(v_t) \mid w_u^t = w_{t,i,u}] \geq (1-1/e) \sum_{(v_t,u)\in E} w_{t,i,u} y_{t,i,u}.$$

Again, the proof carries out similarly to before, using our conditional version of Lemma 3.5, and replacing occurrences of $w_{(v_t,u)}$ and $x_{(v_t,u)}$ with $w_{t,i,u}$ and $y_{t,i,u}$, respectively.

4 Edge Arrivals

4.1 The Algorithm

Similar to the vertex arrival version, we start with an LP for the online problem, and use its solution to build our matching. Again, for each edge $e \in E$, we have the variable x_e represent the probability of e joining the matching in OPT_{on}.

$$\max_{\mathbf{x}} \quad \sum_{e\in E} w_e x_e\,, \tag{20}$$

$$\text{s.t.} \quad \sum_{e \ni u} x_e \leq 1 \qquad \forall\, u \in A \cup B\,, \tag{21}$$

$$p_e \cdot (1 - \sum_{e' \ni v, e' < e} x_{e'}) \geq x_e \qquad \forall\, e = (v, u) \in E\,, \tag{22}$$

$$p_e \cdot (1 - \sum_{e' \ni u, e' < e} x_{e'}) \geq x_e \qquad \forall\, e = (v, u) \in E\,, \tag{23}$$

$$x_e \geq 0 \qquad \forall e \in E\,. \tag{24}$$

We would again like to assert that any x_e corresponding to the execution of OPT_{on} yields a valid solution to this LP. Constraint 21 is as before.

We now consider Constraint 22. In order for OPT_{on} to add $e = (v, u)$ to the matching, it cannot have matched any edge to v already. This occurs with probability exactly $\sum_{e' \ni v, e' < v} x_{e'}$, by definition of e', and since the corresponding events are disjoint. Finally, since p_e being realized is independent from all previous realizations (and any randomness used by the algorithm), the probability that v has not been matched *and* e is realized is given by the left-hand side of Constraint 22, and so the bound must follow. Constraint 23 is similar, and we obtain an observation analogous to Observation 3.1.

Observation 4.1. *Let* $\mathbf{x}$ *be an optimal solution of the LP. We have* $\text{OPT}_{LP} \geq \text{OPT}_{on}$ *where* $\text{OPT}_{LP} = \sum_{e \in E} w_e x_e$.

4.2 The Rounding Procedure

We give our online rounding procedure in Algorithm 2. Here, we think of the vertices $u \in B$ as again making proposals to their neighbours $v \in A$ with some probability (based on b), as long as the corresponding edge e_i is realized. Then, v must decide if it accepts a proposal online. This is as opposed to the vertex arrival model, where v knew all its proposals upon arrival. Since the graph is weighted, simply accepting the first proposal may result in a significant loss. To resolve this issue, our algorithm is designed in a way that each edge e joins the final matching with probability exactly $x_e/2$. In this sense, our algorithm resembles the one designed by Ezra et al. [11] for the vertex arrival version of the problem.

Before stating the algorithm formally, we give a brief overview. The algorithm starts with all the vertices marked as alive, but as the algorithm proceeds it marks some of them as dead. Vertices in B only die when they are matched. However, we sometimes mark a vertex in A as dead without it being matched. At any time t, when edge $e = (v, u)$ arrives, the algorithm needs to decide whether to add this edge to the matching. If v is alive at this point, independent of the status of u, it randomly (with a probability set in Line 8 of the algorithm) decides whether to send a proposal to u. The probability of this event is set in a way that the probability of u ever receiving a proposal from v is equal to x_e. If u is alive, it decides randomly (with a probability set in Line 10 of the algorithm) whether to accept the proposal. If a match happens, we mark u as dead to ensure that we do not match it again in the future. However, vertex v dies iff it send a proposal regardless of the proposal being accepted. This serves two purposes. First, to ensure that its future edges are not matched with a probability higher than $1/2$. Second, to ensure that alive/dead status of the vertices in A are independent of each other throughout the algorithm.

Algorithm 2. Rounding Procedure

1: Let $\mathbf{x}$ be an optimal solution of the LP.
2: Let $M \leftarrow \emptyset$ be a matching of E.
3: Mark all the vertices in V as alive.
4: **for** $t \in |E|$ **do**
5: Let $e_t = (v, u)$ where $v \in A$ and $u \in B$.
6: Define $\alpha_u = \sum_{e \ni u, e < e_t} x_e$.
7: Define $\alpha_v = \sum_{e \ni v, e < e_t} x_e$.
8: Let b be a Bernoulli random variable which is equal to one with probability $\frac{x_{e_t}}{p_{e_t}(1-\alpha_u)}$.
9: **if** u is alive, e_t is realized, and $b = 1$ **then**
10: If v is also alive, then with probability $\frac{1}{2-\alpha_v}$ add e_t to M and mark v as dead.
11: Mark u as dead.
12: **end if**
13: **end for**
14: Return matching M.

We note that this algorithm necessarily returns a valid matching since whenever we add an edge $e_i = (v, u)$ to M, we also mark both v and u as dead (and will never again add any of their incident edges to M). Otherwise, everything is well-defined (notably, $\alpha_v, \alpha_u \in [0, 1]$) by the LP constraints.

4.3 The Analysis

The first half of our analysis will focus on showing that the proposals arriving at a given vertex u are well-behaved. To begin, we show that a vertex $u \in B$ proposes to v with probability exactly $x_{(v,u)}$.

Lemma 4.2. *On any iteration t of Algorithm 2, the probability that the condition at Line 9 holds is x_{e_t}.*

Proof. We prove this by strong induction for a given vertex $u \in A$. Fix $t \geq 1$, and suppose this holds for all $t' < t$. That is, for every $(v, u_{t'}) < t$, the probability that the condition at Line 9 holds (that is, the probability that u proposes to $v_{t'}$) is $x_{e_{t'}}$. Then, defining $\alpha_{t,u} := \sum_{u \ni e, e < (v_t, u)} x_e$ as computed at Line 6, the probability that u is dead at the start of iteration t is exactly $\alpha_{t,u}$, since u is marked dead as soon as it makes its first (and thus only) proposal.

Whether e_t is realized and whether $b = 1$ at iteration t both occur independently of what has occurred so far, and with probabilities $\frac{x_{e_t}}{p_{e_t}(1-\alpha_{t,u})}$ and p_{e_t} respectively. Thus, the probability that all three conditions from Line 9, and that u proposes to v_t, is exactly x_{e_t}. □

Next, we observe that for a fixed $v \in A$, the proposals received from its neighbors $u \in B$ are independent.

Lemma 4.3. *For an edge $e_t \in E$, let P_{e_t} denote the event that in iteration t, the condition at Line 9 holds. Then for any $v \in A$, the events $\{P_{(v,u)} : (v, u) \in E\}$ are independent.*

Proof. Let $e_t = (v, u)$. We observe that $P_{(v,u)}$ depends only on randomness from iteration t (whether e_t was realized, the value of b), as well as whether u is alive or not. However, the aliveness of u itself depends on these same variables from the previous edge incident to u processed by the algorithm (or u is deterministically alive if e_t is the first such edge). Thus, inducting over all such edges,

whether u is alive or not depends only on realizations of edges and values of b from iterations t' where $e_{t'} = (v', u)$ for some v'. Importantly, $P_{(v,u)}$ is a deterministic function of these random inputs, which are importantly disjoint from $P_{(v,u')}$ for other $u' \neq u$. □

Now that we know that the proposals are well-behaved, we can prove our main result for edge arrivals.

Theorem 2. *Algorithm 2 outputs a 1/2-approximate matching, that is*

$$\sum_{e \in E} w_e \mathbb{E}[\mathcal{WM}(e)] \geq \text{OPT}_{on}/2.$$

Proof. We have already noted in Section 4.2 that Algorithm 2 outputs a correct matching. It thus suffices to prove that each edge $e = (v, u)$ is added to M with probability $x_e/2$, by Observation 4.1. For a given v, we prove this by induction over all edges incident to v, in order of arrival.

Fix some $t \geq 1$ where $e_t = (v, u)$. Suppose that any $e_{t'} = (v, u')$ with $t' < t$ is added with probability $x_{e_{t'}}/2$. Then, at time t, the probability that v is already marked dead (equivalently, that an edge incident to v has been added to M) is exactly $\sum_{e \ni v, e<t} x_e/2 = \alpha_v/2$, as these are disjoint events. By Lemma 4.3, the proposals to v were independent, so even conditioned on v being still alive, the probability that v receives a proposal from u is as given by Lemma 4.2, namely x_e. Thus, the probability that v is alive and proposed to by u, and (v, u) is then added to M, is

$$(1 - \alpha_v/2) \cdot x_e \cdot \frac{1}{2 - \alpha_v} = x_e/2.$$ □

5 Proof of Lemma 3.2

Proof. We begin by considering the events E and F. Throughout, we assume a single fixed t, and drop it from the subscripts. First, for any $X \subset B$, the set of events $E^{X \setminus Y} \cap F^Y$ over $Y \subset X$ partition the probability space. In particular, we get the identity

$$\sum_{Y \subset X} \Pr\left[E^{X \setminus Y} \cap F^Y\right] = 1.$$

Even better, this same identity holds when we condition all probabilities by an arbitrary event, so for $X \subset S \subset B$ we have

$$\sum_{Y \subset X} \Pr\left[E^{S \setminus Y} \cap F^Y\right] = \sum_{Y \subset X} \Pr\left[E^{S \setminus X}\right] \Pr\left[E^{X \setminus Y} \cap F^Y \mid E^{S \setminus X}\right] = \Pr\left[E^{S \setminus X}\right] \tag{25}$$

by conditioning on $E^{S \setminus X}$.

Letting $S \subset B$ and $w \in \mathbb{R}^S$ be arbitrary, we now get

$$\begin{aligned}
&\sum_{X \subset S} \Pr\left[E^{S \setminus X}\right] \left(\prod_{u \in X} w_u\right) \prod_{u \in S \setminus X} (1 - w_u) \\
&= \sum_{X \subset S} \left(\sum_{Y \subset X} \Pr\left[E^{S \setminus Y} \cap F^Y\right]\right) \left(\prod_{u \in X} w_u\right) \prod_{u \in S \setminus X} (1 - w_u) && \text{(Equation 25)} \\
&= \sum_{Y \subset S} \Pr\left[E^{S \setminus Y} \cap F^Y\right] \sum_{Y \subset X \subset S} \left(\prod_{u \in X} w_u\right) \prod_{u \in S \setminus X} (1 - w_u)
\end{aligned}$$

$$= \sum_{Y \subset S} \Pr\Big[E^{S\setminus Y} \cap F^{Y}\Big] \left(\prod_{u \in Y} w_u\right) \sum_{Y \subset X \subset S} \left(\prod_{u \in X\setminus Y} w_u\right) \prod_{u \in S\setminus X} (1 - w_u)$$

$$= \sum_{Y \subset S} \Pr\Big[E^{S\setminus Y} \cap F^{Y}\Big] \left(\prod_{u \in Y} w_u\right) \prod_{u \in S\setminus Y} (w_u + (1 - w_u))$$

$$= \sum_{Y \subset S} \Pr\Big[E^{S\setminus Y} \cap F^{Y}\Big] \left(\prod_{u \in Y} w_u\right). \qquad \square$$

References

[1] Saeed Alaei, MohammadTaghi Hajiaghayi, and Vahid Liaghat. Online prophet-inequality matching with applications to ad allocation. In Boi Faltings, Kevin Leyton-Brown, and Panos Ipeirotis, editors, *Proceedings of the 13th ACM Conference on Electronic Commerce, EC 2012, Valencia, Spain, June 4-8, 2012*, pages 18–35. ACM, 2012.

[2] 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.

[3] Soheil Behnezhad and Mahsa Derakhshan. Stochastic weighted matching:(1-ε) approximation. In Sandy Irani, editor, *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 1392–1403. IEEE, 2020.

[4] Soheil Behnezhad, Mahsa Derakhshan, and MohammadTaghi Hajiaghayi. Stochastic matching with few queries: (1-ε) approximation. In Konstantin Makarychev, Yury Makarychev, Madhur Tulsiani, Gautam Kamath, and Julia Chuzhoy, editors, *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, pages 1111–1124. ACM, 2020.

[5] Soheil Behnezhad and Nima Reyhani. Almost optimal stochastic weighted matching with few queries. In Éva Tardos, Edith Elkind, and Rakesh Vohra, editors, *Proceedings of the 2018 ACM Conference on Economics and Computation, Ithaca, NY, USA, June 18-22, 2018*, pages 235–249. ACM, 2018.

[6] Niv Buchbinder, Danny Segev, and Yevgeny Tkach. Online algorithms for maximum cardinality matching with edge arrivals. *Algorithmica*, 81(5):1781–1799, 2019.

[7] Ning Chen, Nicole Immorlica, Anna R. Karlin, Mohammad Mahdian, and Atri Rudra. Approximating matches made in heaven. In Susanne Albers, Alberto Marchetti-Spaccamela, Yossi Matias, Sotiris E. Nikoletseas, and Wolfgang Thomas, editors, *Automata, Languages and Programming, 36th International Colloquium, ICALP 2009, Rhodes, Greece, July 5-12, 2009, Proceedings, Part I*, volume 5555 of *Lecture Notes in Computer Science*, pages 266–278. Springer, 2009.

[8] José R. Correa, Patricio Foncea, Ruben Hoeksma, Tim Oosterwijk, and Tjark Vredeveld. Recent developments in prophet inequalities. *SIGecom Exch.*, 17(1):61–70, 2018.

[9] Kevin P. Costello, Prasad Tetali, and Pushkar Tripathi. Stochastic matching with commitment. In Artur Czumaj, Kurt Mehlhorn, Andrew M. Pitts, and Roger Wattenhofer, editors, *Automata, Languages, and Programming - 39th International Colloquium, ICALP 2012, Warwick, UK, July 9-13, 2012, Proceedings, Part I*, volume 7391 of *Lecture Notes in Computer Science*, pages 822–833. Springer, 2012.

[10] Paul Dütting, Michal Feldman, Thomas Kesselheim, and Brendan Lucier. Prophet inequalities made easy: Stochastic optimization by pricing nonstochastic inputs. *SIAM J. Comput.*, 49(3):540–582, 2020.

[11] Tomer Ezra, Michal Feldman, Nick Gravin, and Zhihao Gavin Tang. Online stochastic max-weight matching: Prophet inequality for vertex and edge arrival models. In Péter Biró, Jason D. Hartline, Michael Ostrovsky, and Ariel D. Procaccia, editors, *EC '20: The 21st ACM*

Conference on Economics and Computation, Virtual Event, Hungary, July 13-17, 2020, pages 769–787. ACM, 2020.

[12] Michal Feldman, Nick Gravin, and Brendan Lucier. Combinatorial auctions via posted prices. In Piotr Indyk, editor, *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 123–135. SIAM, 2015.

[13] Moran Feldman, Ola Svensson, and Rico Zenklusen. Online contention resolution schemes with applications to bayesian selection problems. *SIAM J. Comput.*, 50(2):255–300, 2021.

[14] Buddhima Gamlath, Sagar Kale, and Ola Svensson. Beating greedy for stochastic bipartite matching. In Timothy M. Chan, editor, *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 2841–2854. SIAM, 2019.

[15] Buddhima Gamlath, Michael Kapralov, Andreas Maggiori, Ola Svensson, and David Wajc. Online matching with general arrivals. In David Zuckerman, editor, *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, Baltimore, Maryland, USA, November 9-12, 2019*, pages 26–37. IEEE Computer Society, 2019.

[16] Nick Gravin, Zhihao Gavin Tang, and Kangning Wang. Online stochastic matching with edge arrivals. In Nikhil Bansal, Emanuela Merelli, and James Worrell, editors, *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021, July 12-16, 2021, Glasgow, Scotland (Virtual Conference)*, volume 198 of *LIPIcs*, pages 74:1–74:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[17] Nikolai Gravin and Hongao Wang. Prophet inequality for bipartite matching: Merits of being simple and non adaptive. In Anna Karlin, Nicole Immorlica, and Ramesh Johari, editors, *Proceedings of the 2019 ACM Conference on Economics and Computation, EC 2019, Phoenix, AZ, USA, June 24-28, 2019*, pages 93–109. ACM, 2019.

[18] Mohammad Taghi Hajiaghayi, Robert D. Kleinberg, and Tuomas Sandholm. Automated online mechanism design and prophet inequalities. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, July 22-26, 2007, Vancouver, British Columbia, Canada*, pages 58–65. AAAI Press, 2007.

[19] Richard M. Karp, Umesh V. Vazirani, and Vijay V. Vazirani. An optimal algorithm for on-line bipartite matching. In Harriet Ortiz, editor, *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, May 13-17, 1990, Baltimore, Maryland, USA*, pages 352–358. ACM, 1990.

[20] Robert Kleinberg and S. Matthew Weinberg. Matroid prophet inequalities and applications to multi-dimensional mechanism design. *Games Econ. Behav.*, 113:97–115, 2019.

[21] Ulrich Krengel and Louis Sucheston. On semiamarts, amarts, and processes with finite value. *Probability on Banach spaces*, 4:197–266, 1978.

[22] Brendan Lucier. An economic view of prophet inequalities. *SIGecom Exch.*, 16(1):24–47, 2017.

[23] Christos H. Papadimitriou, Tristan Pollner, Amin Saberi, and David Wajc. Online stochastic max-weight bipartite matching: Beyond prophet inequalities. In Péter Biró, Shuchi Chawla,

and Federico Echenique, editors, *EC '21: The 22nd ACM Conference on Economics and Computation, Budapest, Hungary, July 18-23, 2021*, pages 763–764. ACM, 2021.

[24] Amin Saberi and David Wajc. The greedy algorithm is not optimal for on-line edge coloring. In Nikhil Bansal, Emanuela Merelli, and James Worrell, editors, *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021, July 12-16, 2021, Glasgow, Scotland (Virtual Conference)*, volume 198 of *LIPIcs*, pages 109:1–109:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[25] Alfredo Torrico and Alejandro Toriello. Dynamic relaxations for online bipartite matching. *arXiv preprint arXiv:1709.01557*, 2017.

[26] Yutaro Yamaguchi and Takanori Maehara. Stochastic packing integer programs with few queries. In Artur Czumaj, editor, *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 293–310. SIAM, 2018.