Operations Research

Online Matching Frameworks Under Stochastic Rewards, Product Ranking, and Unknown Patience

Brian Brubach, Nathaniel Grammel, Will Ma, Aravind Srinivasan

https://pubsonline.informs.org/journal/opre

OPERATIONS RESEARCH
Articles in Advance, pp. 1–16
ISSN 0030-364X (print), ISSN 1526-5463 (online)

Methods

Online Matching Frameworks Under Stochastic Rewards, Product Ranking, and Unknown Patience

Brian Brubach,[a] Nathaniel Grammel,[b] Will Ma,[c,*] Aravind Srinivasan[b]

[a] Computer Science Department, Wellesley College, Wellesley, Massachusetts 02481; [b] Department of Computer Science, University of Maryland, College Park, Maryland 20742; [c] Graduate School of Business and Data Science Institute, Columbia University, New York, New York 10027

*Corresponding author

Contact: bb100@wellesley.edu (BB); ngrammel@umd.edu (NG); wm2428@gsb.columbia.edu, https://orcid.org/0000-0002-2420-4468 (WM); asriniv1@umd.edu, https://orcid.org/0000-0002-3409-6077 (AS)

Received: June 9, 2021
Revised: July 22, 2022; June 26, 2023
Accepted: September 25, 2023
Published Online in Articles in Advance: October 27, 2023

Area of Review: Optimization

https://doi.org/10.1287/opre.2021.0371

Copyright: © 2023 INFORMS

Abstract. We study generalizations of online bipartite matching in which each arriving vertex (customer) views a ranked list of offline vertices (products) and matches to (purchases) the first one they deem acceptable. The number of products that the customer has patience to view can be stochastic and dependent on the products seen. We develop a framework that views the interaction with each customer as an abstract resource consumption process and derive new results for these online matching problems under the adversarial, nonstationary, and independent and identically-distributed arrival models, assuming we can (approximately) solve the product ranking problem for each single customer. To that end, we show new results for product ranking under two cascade-click models: an optimal algorithm when each item has its own hazard rate for making the customer depart and a 1/2-approximate algorithm when the customer has a general item-independent patience distribution. We also present a constant-factor 0.027-approximate algorithm in a new model where items are not initially available and arrive over time. We complement these positive results by presenting three additional negative results relating to these problems.

Funding: N. Grammel was supported in part by NSF award [CCF-1918749] and by research awards from Amazon and Google. A. Srinivasan was supported in part by NSF awards [CCF-1422569, CCF-1749864, and CCF-1918749], as well as research awards from Adobe, Amazon, and Google. W. Ma was supported in part by a research award from Amazon.
Supplemental Material: The e-companion is available at https://doi.org/10.1287/opre.2021.0371.

Keywords: online algorithms • competitive ratio • online matching • product ranking

1. Introduction

Online matching is a fundamental problem in e-commerce and online advertising, introduced in the seminal work of Karp et al. (1990). While offline matching has a long history in economics and computer science, online matching has exploded in popularity with the ubiquity of the internet and the emergence of online marketplaces. A common scenario in e-commerce is the online sale of unique goods because of the ability to reach niche markets via the internet (e.g., eBay); typical products include rare books, trading cards, art, crafts, and memorabilia. We will use this as a motivating example to describe our setting. However, the settings we study can also model job search/hiring, crowdsourcing, online advertising, ride-sharing, and other online matching problems.

In classical online bipartite matching, we start with a known set of *offline vertices* that may represent items for sale or ads to be allocated. Then, an unknown sequence of *online vertices* arrives, which may represent customers, users, or visitors to a web page. These online vertices or customers arrive one by one, and the decision to match each customer or not (and if so, to which item) must be made irrevocably before the next customer is revealed. In the original formulation, the online vertices are chosen fully adversarially, although models that assume they are drawn from probability distributions have since been studied (Feldman et al. 2009, Alaei et al. 2012).

Many generalizations of online matching have also been proposed, including stochastic rewards and weighted graphs. Under stochastic rewards, there can be repeated interactions with a customer (recommending an item, and if they do not accept, recommending another item, etc.) before the next one arrives, as we describe subsequently. Our paper's goal is to provide a framework that decouples the repeated-interaction problem for a single customer from the overall allocation problem over multiple customers, leading to new results and unification of old ones.

Moreover, motivated by product ranking, we derive new results for the repeated-interaction problem with a single customer, including the extension in which the horizon for these interactions is unknown or stochastic.

1.1. Description of Stochastic Rewards Model, with Patience

In the *stochastic rewards* model, each edge exists independently according to a known probability; this probability is revealed upon the arrival of its incident online vertex. This is motivated by online platforms in which only a probabilistic prediction of whether a customer will buy an item is known at the time they arrive. The algorithm can "probe" edges incident to an online vertex, or equivalently recommend the customer an item, after which if they accept, then the item is sold committedly. If they otherwise reject, then under the basic stochastic rewards model (Mehta and Panigrahi 2012) there is no opportunity to offer another item; this is known as the customer having a *patience* of 1.

Other papers (Bansal et al. 2010, Adamczyk et al. 2015, Brubach et al. 2017) have more generally allowed the customer to have any deterministic patience θ. This can be interpreted as a *product ranking* problem where θ different items are listed on a page, and the customer will view them in order, stopping once they see an acceptable item or reaching the end of the page. The product ranking problem where θ is deterministic can be efficiently solved using dynamic programming (Purohit et al. 2019). More generally, if θ is random but drawn from a known distribution, then the customer may probabilistically depart after seeing any undesirable item; this is called the *cascade-click* model of product ranking. We will derive new results for the cascade-click model.

1.2. Description of Edge Weights and Stochastic Arrival Models

In an orthogonal generalization of online bipartite matching, edges between items and customers may have a *weight*, which is the reward collected when that edge is matched. This can represent, for example, the price at which that item is sold to the customer. When edges can take on different possible weights, parametric competitive ratios are known (Ma and Simchi-Levi 2020), but a competitive ratio that is an absolute constant is impossible in the original adversarial arrival model (see Mehta 2012). Therefore, many papers have focused on the relaxed models of *stochastic arrivals* or *vertex weights* instead, each of which circumvents this impossibility.

In the stochastic arrival models, the total number of online vertices T is known, and each online vertex $t = 1, \ldots, T$ has a *type*[1] drawn independently from a known distribution. Generally, we allow distributions to be *nonstationary* and vary with t, although we also consider the IID special case where these distributions are identical. Stochastic arrival models are motivated by settings with sufficient data to estimate the distribution over types. Meanwhile, in the model with vertex weights, all edges incident to any offline vertex u must have the same weight. This is motivated by each offline item having its own fixed price that is identical across customers.

1.3. Our Contributions

We develop a decoupling framework, which we describe in greater detail in Section 1.3.1, wherein we first study a simpler, single-customer version of various stochastic matching problems and then use the algorithms for these problems to inform decisions during online customer arrivals. This approach allows us to derive new results for online bipartite matching with stochastic rewards and, in many cases, stochastic patience as well.

We consider both vertex weights and general edge weights in combination with the adversarial, nonstationary, and IID arrival models. Since our framework requires the repeated-interaction problems to be solvable for a single customer, we also make advancements on this front (see Section 1.3.2): namely, improving algorithms for the *cascade-click* model of product ranking, and deriving new algorithms in a model where *items are arriving* over time. Finally, we derive several negative results of interest (see Section 1.3.3).

1.3.1. Framework That Decouples Online Matching from Single-Customer Problems.

First, we build a framework that takes as input a subroutine for solving the single-customer problem, and outputs an algorithm for the overall multicustomer online matching problem, under the variants we mentioned. The competitive ratios guaranteed by our framework are explained in Table 1, and we would like to highlight a key distinction in our approach. Existing analyses of stochastic rewards (Bansal et al. 2010, Mehta and Panigrahi 2012, Adamczyk et al. 2015, Brubach et al. 2017) all use a linear program (LP) that is *specific* to the stochastic rewards matching process, which exhibits a stochasticity gap (see Section 6.1). By contrast, our framework uses an *abstract* LP, in which

- There is a variable $x_v(\pi)$ for each of the (exponentially many) policies π that could be used for interacting with a single customer of type v;
- Each such policy π ends up matching each available offline vertex u with probability $p_{uv}(\pi)$;
- There is a single set of constraints tying together the customers over time, which enforce that each offline vertex u is matched at most once in expectation.

Our framework abstracts away the details of the stochastic rewards matching process, deterministic versus stochastic patience, etc., and holds as long as the policies represent different consumption processes[2] that use up the offline vertices u *independently* over time according to known probabilities.

Our results, however, are predicated on the existence of a subroutine that can (approximately) solve the

Table 1. Landscape of Online Matching Results

	Unweighted	Vertex weighted	Edge weighted
Adversarial			
Nonstochastic	0.632 (tight) (Karp et al. 1990)	0.632 (tight) (Aggarwal et al. 2011)	[must be weight dependent] (Ma and Simchi-Levi 2020)
Stochastic rewards	0.5 (Mehta and Panigrahi 2012)	? → **0.5**	[must be weight dependent] (Ma and Simchi-Levi 2020)
Deterministic patience/hazard rate model	? → **0.5**	? → **0.5**	–
Stochastic patience	? → **0.25**	? → **0.25**	–
Nonstationary			
Nonstochastic	0.632 (Alaei et al. 2012)	0.632 (Alaei et al. 2012)	0.5 (Alaei et al. 2012)
Deterministic patience/hazard rate model	? → **0.632**	? → **0.632**	? → **0.5**
Stochastic patience	? → **0.316**	? → **0.316**	? → **0.25**
Known IID			
Nonstochastic	0.729 (Brubach et al. 2020)	0.729 (Brubach et al. 2020)	0.705 (Brubach et al. 2020)
Stochastic rewards	0.632 (Brubach et al. 2020)	0.632 (Brubach et al. 2020)	0.632 (Brubach et al. 2020)
Deterministic patience/hazard rate model	0.46 → **0.632** (Brubach et al. 2017)	0.46 → **0.632** (Brubach et al. 2017)	0.46 → **0.632** (Brubach et al. 2017)
Stochastic patience	? → **0.316**	? → **0.316**	? → **0.316**

Notes. Landscape of online matching results grouped by arrival model and form of edge weights, and including the unknown patience models we introduce: the (item-dependent) hazard rate model and the arbitrary (item-independent) stochastic patience model. Bold results with arrows show the improvements from this paper, with question marks denoting problems where no prior bound was known.

repeated-interaction problems for each customer. We will call such a subroutine κ*-approximate* if, given any weights w_{uv}, it finds a policy π whose immediate expected reward $\sum_u w_{uv} p_{uv}(\pi)$ is at least κ times the maximum possible immediate expected reward over all policies, for some $\kappa \in [0, 1]$. Equipped with a κ-approximate subroutine, our framework provides

1. A $\kappa/2$-competitive algorithm for vertex weights and adversarial arrivals (Section 4.1);
2. A $\kappa/2$-competitive algorithm for edge weights and nonstationary arrivals (Section 4.2);
3. A $(1 - 1/\mathrm{e})\kappa$-competitive algorithm for edge weights and IID arrivals (Section 4.3); and
4. A $(1 - 1/\mathrm{e})\kappa$-competitive algorithm for vertex weights and nonstationary arrivals (Section 4.4).

The value $\kappa = 1$ is possible when θ is deterministic (Purohit et al. 2019). We derive new results showing that $\kappa = 1$ is also possible when θ follows an (item-dependent) hazard rate model, and that $\kappa = 1/2$ is possible when θ follows any (item-independent) distribution. This, in conjunction with our framework, justifies all of the results in Table 1.

1.3.2. New κ-Approximate Subroutines for Single-Customer Problems.

As discussed, it is important for our framework to have κ-approximate subroutines for the repeated-interaction problems with a single customer. We make the following advancements on this front:

1. A 1-approximate (optimal) subroutine, in the model where each item i has a known *hazard rate* r_i and, if seen by the customer and undesired, causes the customer to depart with probability (w.p.) r_i;
2. A 1/2-approximate subroutine, in the model where the customer has an arbitrary known patience distribution (and the probabilities of departing do not depend on the items seen);
3. A 0.027-approximate subroutine, in a new model where the customer has a deterministic patience, but the items are arriving over time according to Bernoulli processes.

The first two models can be motivated by product ranking in e-commerce. A special case of the first model (Section 5.1) is where r_i is equal to some r for all i, which represents a patience distribution with *constant* hazard rate r, that is, a customer who departs w.p. r after each position regardless of the item seen. Meanwhile, our 1/2-approximation for the second model (Section 5.2) improves the state-of-the-art $1/e$-approximation from Chen et al. (2021) for this cascade-click model of product ranking. Their result also only holds in the special case of increasing hazard rate, while we extend it to general distributions by formulating and rounding a new LP relaxation[3] for this single-customer problem. We note that general patience distributions are well motivated in applications; see, for example, Aveklouris et al. (2021), who study a matching model where items are also arriving over time. On that note, our result for the third model (Section 5.3), when plugged into our frameworks, provides constant-factor guarantees in a related model where items (representing contractors in an online labor platform) may not be present at the beginning and need to arrive online after each customer (to acknowledge they can perform the customer's task), and the customer has to then also accept that contractor. We contrast this new model with other online platform matching models in Section 2.

1.3.3. Negative Results. Finally, our work presents three important negative results:

1. We formalize the notion of a stochasticity gap for LP-based approaches to these problems and construct a stochastic bipartite graph in which even the offline maximum matching has an expected size of at most 0.544 times the value of the LP relaxation (Section 6.1). This means that the competitive ratio from the existing LP-based approaches cannot be better than 0.544, while our framework yields a $1-1/\mathrm{e} \approx 0.632$-competitive algorithm.
2. We show that the simple family of greedy algorithms introduced in Mehta and Panigrahi (2012) cannot be better than $1/2$-competitive (Section 6.2).
3. We show that when offering items to a single customer with random patience, if one compares to a benchmark that knows the realization of the patience in advance, then any constant-factor approximation is impossible (Section 6.3). Importantly, our counterexample holds even if the customer can be repeatedly offered the same item, which is identical to having an unknown number of opportunities to make a *single* sale (since the customer will buy at most one item). This is similar in spirit to the negative result derived in Alijani et al. (2020).

2. Further Related Work

2.1. Online Matching with Stochastic Rewards

Online matching represents a large literature, which has been surveyed in Mehta (2012). We will describe the portion of this literature that focuses on stochastic rewards, where edges only match probabilistically upon being probed. This problem has been studied under both adversarial and stochastic arrival models, as well as different variants depending on the assumptions about edge weights/patience.

Online matching with stochastic edges was introduced in Bansal et al. (2010)[4] as stochastic matching with timeouts (patience), where the authors showed a ratio of 0.12 for known IID arrivals and arbitrary edge weights. This was later improved to 0.46 in Brubach et al. (2017)[5] and to 0.51 in Fata et al. (2019) for some cases. We improve these results by establishing a competitive ratio of $1/2$ for nonstationary arrivals and $1-1/\mathrm{e}$ for IID arrivals. We note that Borodin et al. (2022) concurrently prove these results, differing in three ways: (i) They allow for more general constraints on which edges can be probed, beyond a simple patience constraint (although they do not consider stochastic patience). (ii) They compare against a more powerful offline benchmark that can switch back and forth between probing different online vertices. (iii) They show that $1-1/\mathrm{e}$ holds in the more general model of nonidentical independent draws arriving in a uniformly random order. The same authors have also studied online matching with stochastic edges under the "secretary" model of random-order arrival (see Borodin et al. 2021).

For adversarial arrivals, most work has focused on the unweighted case, initially studied by Mehta and Panigrahi (2012) in the special case where patience θ_v equals one for all v. Under the further restriction of uniform vanishing edge probabilities, they showed that a competitive ratio of 0.53 is possible. This was extended to a ratio of 0.534 for unequal, but still vanishingly small, probabilities (Mehta et al. 2015). These results were also recently improved to 0.576 and 0.572, respectively, by Huang and Zhang (2020) and then to 0.596 for both models by Goyal and Udwani (2023); however, all these results focus on the case of vanishingly small probabilities, do not consider patience values greater than one, and do not consider vertex weights. For arbitrary edge probabilities, general deterministic patience values, and vertex weights, our guarantee of 0.5 is the best known. There is also a hardness result in Mehta and Panigrahi (2012) which shows that no algorithm for stochastic rewards with adversarial arrivals can achieve a competitive ratio greater than 0.62. This quantity is strictly less than $1-1/\mathrm{e}$, although we argue that this difference is artificially caused by the stochasticity gap, as we explain in Section 6.1.

Golrezaei et al. (2014) study another model of stochastic rewards, in which when a vertex v (viewed as a customer) arrives online, an online algorithm chooses a *set* S of potential matches for v (viewed as an offering of products to the customer). Each customer (online vertex) has a *general choice model* which specifies the probability of the customer purchasing each item when offered each possible set of product assortments S. We contrast this model in more detail in Online Appendix B, but note that in this setting, a set of potential matches is chosen *all at once* rather than probed sequentially, with the outcome being determined by full set S (the offered product assortment).

2.2. Large Starting Capacities

We do not study how our guarantees improve if there are at least k copies of every offline vertex, although we believe our frameworks could be expanded to do so. The state of the art for these k-dependent guarantees in online matching can be found for general adversarial arrivals (Ma and Simchi-Levi 2020), unweighted adversarial arrivals (Kalyanasundaram and Pruhs 2000), general nonstationary arrivals (Jiang et al. 2022), nonstationary arrivals with vertex weights (Alaei et al. 2012), and IID arrivals (Ma et al. 2021).

2.3. Cascade-Click Models in Product Ranking

We turn our literature review to papers that study the repeated-interaction/product ranking problems for a single customer. Our result from Section 5.1 shows how to optimally solve this problem under constant hazard rate, a special case of interest in Chen et al. (2021). Our result in Section 5.2 improves their guarantee and holds for general patience distributions. We should note that our results do not directly apply to more general cascade-click models (see Kempe and Mahdian 2008) where the probability of the customer running out of patience depends on the specific item shown, but we believe that our simple LP-based technique in Section 5.2 could be useful for these generalized models. Other generalized ranking problems involving choice models are studied in Derakhshan et al. (2022).

In the related *sequential assortment* problem, multiple products can be shown to the customer at a time. The customer chooses between them according to a Multinomial Logit (MNL) choice model (instead of independent click probabilities), and alternatively the customer could choose the option of viewing the next assortment, never to return. There is a constraint that the same product cannot be shown in different assortments. This problem is typically studied when the number of stages is deterministic and known (see Feldman and Segev 2019 and the references therein); however, the *stage-dependent* coefficients in Feldman and Segev (2019) can be used to capture our notion of a stochastic patience. Nonetheless, the polynomial-time approximation scheme (PTAS) derived in Feldman and Segev (2019) does not subsume our $1/2$-approximation because in the sequential assortment problem there is no constraint on the number of products offered at once, hence it does not capture our problem; also, in a PTAS, to get a $(1-\varepsilon)$-approximation the runtime needs to be exponential in $1/\varepsilon$, whereas our LP-based technique has polynomial runtime independent of any error parameter.

2.4. Online Matching Where Items Arrive over Time

Motivated by online platforms, many models where items arrive over time have been recently studied, with constant-factor approximations (Aouad and Saritaç 2022, Kessel et al. 2022) and optimal algorithms (Aveklouris et al. 2021, Kerimov et al. 2021) known under certain regimes. These papers focus on steady-state behavior, which is possible because items are arriving indefinitely. Our paper contrasts these models because there is still a finite supply of items; they merely need to “arrive” to acknowledge each customer, and we provide a constant-factor approximation for any finite time horizon and market size.

3. Problem Definition and Notation

We use $G=(U,V;E)$ to denote a bipartite graph with vertex set $U\cup V$ and edge set $E\subseteq U\times V$. Let $U=\{u_1,\ldots,u_m\}$ represent offline vertices and $V=\{v_1,\ldots,v_n\}$ represent online vertices. For an edge $e=(u,v)$, we denote the *weight* of edge e by w_e or $w_{u,v}$; for the special case of vertex weights, each offline vertex u_i has a weight denoted by w_i, and $w_{u_i,v}=w_i$ for all $v\in V$. We will generally consider *stochastic edges*, which means that for each edge $(u_i,v_j)\in U\times V$, there is a known probability $p_{i,j}$ with which that edge will independently exist when probed.

When considering the online matching problem for a single online vertex (customer) v, we will refer to it as a *star graph*. In this case, we simplify notation and write p_i to denote the probability of edge (u_i,v). We also use $p_{u,v}$ for the given probability of edge (u,v) when indices i and j are not required. Without loss of generality, we may assume that $p_{u,v}$ is defined even for $(u,v)\notin E$, because in this case we can simply let $p_{u,v}=0$.

We are further given a *patience* value θ_v for each online vertex in V (we may also write θ_j for the patience of vertex $v_j\in V$) that signifies the number of times we are allowed to probe different edges incident on v when it arrives. Each edge may be probed at most once, and if it exists, we must match it and stop probing (probe-commit model).

We consider the online vertices arriving at positive integer *times*. In the adversarial arrival model, the vertices of $V=\{v_1,v_2,\ldots,v_n\}$ are fixed and the order of their arrival is set by an adversary so as to minimize the expected matching weight. We assume without loss of generality that the vertices arrive in the order $v_1,v_2,\ldots,v_n$. When we consider the stochastic arrival models, V instead specifies a set of vertex *types*, and at time t, a vertex of an independently randomly chosen type from a known distribution arrives. Generally, these distributions can vary across time, which we call the *prophet arrival*[6] model; we also consider the special case where these distributions are identical, which we refer to as *IID arrivals*. In these models, we will let T denote the length of the time horizon which is assumed to be known (otherwise the problem is impossible; see Section 6.3).

When an online vertex v arrives at time t, we attempt to match it to an available offline vertex. We are allowed to probe edges incident to v_t one by one, stopping as soon as an edge (u_i,v_t) is found to exist, at which point the edge is included in the matching and we receive a reward of w_i. We are allowed to probe a maximum of θ_t edges (in the stochastic patience models, θ_t is not known a priori and is discovered only after θ_t failed probes); if θ_t edges are probed and none of the edges exist, then vertex v_t remains unmatched and we receive no reward. If we successfully match v_t to u_i, we say that w_i is the *value* or *reward* of v_t's match; if v_t remains unmatched, we say it has a value or reward of zero. The next online vertex v_{t+1} does not arrive until we have finished attempting to match v_t (either by exhausting the patience constraint or by successfully matching v_t). Thus, there is only ever one online vertex available for matching at any one time.

We use G to denote an instance, which includes the graph, weights, edge probabilities, and any arrival distributions including patience. Given any instance G one can consider an optimal *offline* algorithm, which knows in advance the online vertices that will arrive. For the adversarial arrival model, this means knowing the sequence $v_1, v_2, \ldots, v_n$; for the stochastic arrival models, this means knowing the sequence of T types that will be realized. We let $\mathsf{OPT}(G)$ denote the expected reward collected by the best sequential probing algorithm on G that has access to this offline information, noting that (i) $\mathsf{OPT}(G)$ does not know the realizations of the stochastic edges in advance either; (ii) computing $\mathsf{OPT}(G)$ is difficult but unnecessary; (iii) for stochastic arrival models, this expectation is also over the realizations of the T types; and (iv) we assume that the offline algorithm must finish[7] the interactions with one online vertex before moving to the next. Meanwhile, we let $\mathsf{ALG}(G)$ denote the expected reward collected by a fixed online algorithm on G, again taking a realization over types in the stochastic arrival models, and any potential randomness in the algorithm.

With this understanding, we say that a fixed (potentially randomized) online algorithm is *c-competitive* if $\mathsf{ALG}(G)/\mathsf{OPT}(G) \geq c$ for all instances G, where c is a constant in [0,1]. We are interested in the maximum value of c for which an algorithm can be c-competitive, which is referred to as the *competitive ratio*.

3.1. Outline for the Rest of the Paper

Our main algorithms and results for online matching with stochastic edges are presented in Section 4. In that section, we first present an algorithm for the vertex-weighted case, under adversarial arrivals, and show that it is $1/2$-competitive. To our knowledge, this is the first result for this setting. In addition, we provide an algorithm for the *edge-weighted* case, under prophet arrivals. Here, too, we are able to show the algorithm is $1/2$-competitive; we further show that a slight modification can improve the competitive ratio to $1 - 1/\mathsf{e}$ when either the edge-weighted assumption is relaxed to vertex weights or the nonstationary assumption is relaxed to known IID arrivals.

All of the algorithms of Section 4 rely on utilizing, as a black box, an algorithm for the simpler problem of a star graph, which corresponds to a single online customer. For this problem, when the patience of the customer is known, there is an optimal algorithm based on dynamic programming because of Purohit et al. (2019). However, the results in Section 4 are stated in an abstract general manner, which allows us to swap out the algorithm of Purohit et al. (2019) for algorithms solving the star graph problem under different settings of patience. In Section 5, we introduce new, stochastic models for the patience of the customer and give algorithms for these new settings. These new star graph algorithms can then be used as black boxes in the algorithms of Section 4, giving results for online matching under these new patience models.

Finally, in Section 6, we present our negative results following the order described.

4. Algorithms for Online Matching with Stochastic Edges

In this section, we present our results for online matching with stochastic edges. Recall that in this problem, the items U are known in advance while the customers arrive one by one in an online fashion. When a customer of type v arrives, we learn the probability $p_{u,v}$, for each $u \in U$, that the customer will purchase item u if offered; if purchased, we gain some reward specified by $w_{u,v}$, and if not, we may proceed to offer another item up to a total of θ_v offers. In the simplest setting, θ_v is known to the algorithm, although our framework can also handle settings where only a probability distribution over θ_v is known (see Section 5). An online algorithm must offer items sequentially to the customer and must make all its offers before the next one arrives. The goal is to maximize the expected total reward across all customers.

4.1. Vertex-Weighted, Adversarial Arrivals

We present a greedy algorithm, AdvGreedy, which achieves a 0.5-approximation for online matching with vertex weights, stochastic rewards, and patience constraints in the adversarial arrival model. Recall that in this setting, each offline item $u_i \in U$ has a weight w_i such that $w_{u_i,v} = w_i$ for all customers $v \in V$, modeling the situation where items have a fixed price that is the same for all customers. If vertex v is the t^{th} vertex to arrive online, we say *v arrives at time t*. Recall that the goal of the problem is to offer items to customer v_t one by one, until either the patience runs out or an item is successfully sold, and the entirety of this process is carried out before the $(t+1)^{\text{st}}$ arrival, v_{t+1}, arrives. The goal is to maximize the total revenue across all arrivals. Our algorithm makes use of a black box subroutine, StarBB, which takes as input a single online vertex, along with the probabilities and weights of its incident edges; StarBB$(v, \mathbf{p}, \mathbf{w})$ simply probes edges incident to online vertex v in some order, until either v's patience is exhausted or a match is successful. Our results hold as long as StarBB is optimal, or within a constant factor κ of optimal. When the patience is deterministic and known to the algorithm, we can use the dynamic programming-based algorithm of Purohit et al. (2019) for StarBB; the dynamic program gives a sequence of θ_v vertices in U to probe in this order. This is optimal for star graphs, so $\kappa = 1$. In Section 5, we extend this online stochastic matching result to settings where the patience is unknown and stochastic, by giving star graph algorithms for these settings and proving constant-factor approximations for them.

Algorithm 1 (Use a Star Graph Black Box to Greedily Match Arriving Vertices)

```
Function AdvGreedy($U$, $V$, $\mathbf{p}$, $\mathbf{w}$):
    for Arriving vertex $v \in V$ do
        StarBB($v, \mathbf{p}, \mathbf{w}$)
```

Let ALG(G) denote the expected size of the matching produced by this algorithm on the graph G. Let OPT(G) denote the expected size of the matching produced by an optimal *offline* algorithm. Our main result here is

Theorem 1. *Given a κ-approximate black box for solving star graphs, Algorithm 1 achieves a competitive ratio of 0.5κ; that is, for any bipartite graph G, $\frac{\mathsf{ALG}(G)}{\mathsf{OPT}(G)} \geq 0.5\kappa$.*

To prove this, we first present an LP which provides an upper bound on the offline optimal.

4.1.1. An LP Upper Bound on OPT(G). We formulate a new LP for our problem by adding a new constraint, (1d), to the standard LP relaxation of the problem. This new LP gives a tighter upper bound on the offline optimal solution, OPT(G). Note that our algorithm does not need to solve this new LP, as it is only used in our analysis.

$$\mathsf{OPT}_{\mathsf{LP}} := \max \sum_{u\in U}\sum_{v\in V} x_{u,v}p_{u,v}w_u \tag{1}$$

$$\text{subject to} \sum_{v\in V} x_{u,v}p_{u,v} \leq 1 \qquad \forall u \in U \tag{1a}$$

$$\sum_{u\in U} x_{u,v}p_{u,v} \leq 1 \qquad \forall v \in V \tag{1b}$$

$$\sum_{u\in U} x_{u,v} \leq \mathbb{E}[\theta_v] \qquad \forall v \in V \tag{1c}$$

$$\sum_{u\in U'} x_{u,v}p_{u,v}w_u \leq \mathsf{OPT}(U',v) \qquad \forall U' \subseteq U, v \in V \tag{1d}$$

$$0 \leq x_{u,v} \leq 1 \qquad \forall u \in U, v \in V$$

In this LP, we slightly abuse notation and write OPT(U', v) to denote OPT(G') for a star graph $G' = (U', \{v\}, U' \times \{v\})$. Recall that $OPT(U', v)$ can be computed by a black box.

We first show in Online Appendix A that our strengthened LP is a valid upper bound.

Lemma 1. *For any bipartite graph G,* $\mathsf{OPT}_{\mathsf{LP}}(G) \geq \mathsf{OPT}(G)$.

4.1.2. Proof of 0.5-Competitiveness. We can then bound the performance of our greedy algorithm relative to the solution of LP (1).

Lemma 2. *If* StarBB *is a κ-approximate algorithm for star graphs, then for any bipartite graph G,* ALG$(G) \geq 0.5\kappa$ OPT$_{\mathsf{LP}}(G)$.

By Lemmas 1 and 2, we have ALG$(G) \geq 0.5\kappa$OPT$_{\mathsf{LP}}$ $(G) \geq 0.5\kappa$OPT(G), which implies our main result of a $\frac{\kappa}{2}$-competitive algorithm. As mentioned, using the probing order given by the dynamic program of Purohit et al. (2019) as StarBB gives $\kappa = 1$, so we have a $\frac{1}{2}$-approximation. In Section 5, we present star graphs for stochastic patience settings, where κ is not necessarily one.

4.2. Prophet Arrival Setting

If we wish to allow for arbitrary edge weights, we must consider a different arrival model. A popular arrival model in the literature is the known IID setting, as discussed in Section 2; here, we consider a generalization of the known IID setting, which we call the *prophet arrival model*. In this model, V specifies a set of possible arrival *types*; each arrival takes on one of these types randomly, according to a known distribution. The probabilities at each arrival are independent of previous arrivals, and the distribution over possible types can be different at each arrival.

For $t = 1, 2, \ldots, T$ and $v \in V$, denote by q_{tv} the probability that the vertex arriving at time t will be of type v. For convenience, we denote by $q_v = \sum_{t=1}^{T} q_{tv}$ the expected number of arrivals of a vertex of type v.

We employ a new exponential-sized LP relaxation. In this LP, the variables correspond to *policies* for probing an arriving online vertex. A deterministic policy π for matching any online vertex type v is characterized by a permutation of some *subset of* U. The policy specifies the strategy of attempting to match v to vertices of U in the order given by π, until either a probe is successful, all vertices in π are attempted, or the patience of v is exhausted. Let $\mathcal{P}$ denote the set of all deterministic policies.

We present our LP in (2). We let $p_{uv}(\pi)$ denote the probability that an online arrival of type v is matched to offline vertex u when following policy π, assuming that all vertices of π are still unmatched. The decision variables in the LP are given by $x_v(\pi)$, and we let OPT$_{\mathsf{LPP}}(G)$ denote the true optimal objective value of the exponential-sized LP.

$$\mathsf{OPT}_{\mathsf{LPP}} = \max \sum_{v\in V}\sum_{\pi\in\mathcal{P}} x_v(\pi) \sum_{u\in U} p_{uv}(\pi)w_{uv} \tag{2}$$

$$\text{subject to} \sum_{v\in V}\sum_{\pi\in\mathcal{P}} x_v(\pi)p_{uv}(\pi) \leq 1 \quad \forall u \in U \tag{2a}$$

$$\sum_{\pi\in\mathcal{P}} x_v(\pi) = q_v \qquad \forall v \in V \tag{2b}$$

$$x_v(\pi) \geq 0 \qquad \forall v \in V, \pi \in \mathcal{P} \tag{2c}$$

We can interpret $x_v(\pi)$ as the expected number of times policy π will be applied to an online vertex of type v. Constraint (2a) says that in expectation each offline vertex u can be matched at most once. Constraint (2b) comes from the fact that exactly one policy (possibly the policy that makes zero probes) must be applied to each arriving vertex of type v. It follows from standard techniques (see, e.g., Bansal et al. 2010, lemma 9) that even for a clairvoyant, who knows the realized types of arrivals in advance, if we let $x_v(\pi)$ denote the expected number of times it applies policy π on an online vertex of type v, then this forms a feasible solution to the LP with objective value equal to the clairvoyant's expected weight matched. Therefore, if we can bound the algorithm's expected reward relative to $\mathsf{OPT}_{\mathsf{LPP}}(G)$, then this would yield a competitive ratio guarantee.

4.2.1. Solving the Exponential-Sized LP. The LP (2) has a variable $x_v(\pi)$ for every possible online vertex type v and policy $\pi \in \mathcal{P}$. If $\mathcal{P}$ has polynomial size, this LP therefore has polynomial size. This happens, for example, when all online vertices have a small constant patience. For instance, when the patience of all vertices is one, a policy $\pi \in \mathcal{P}$ is characterized by a single offline vertex and so $|\mathcal{P}| = |U|$.

However, for general patience values the LP has exponentially many variables. Nonetheless, since there are only a polynomial number of constraints, a sparse solution which is polynomially sized still exists. To help in solving the LP, we consider the dual.

$$\text{minimize} \sum_{u\in U}\alpha_u + \sum_{v\in V} q_v\beta_v \tag{3}$$

$$\text{s.t.} \sum_{u\in U} p_{uv}(\pi)\alpha_u + \beta_v \geq \sum_{u\in U} p_{uv}(\pi)w_{uv} \qquad \forall v\in V, \pi\in\mathcal{P} \tag{3a}$$

$$\alpha_u \geq 0 \qquad \forall u \in U \tag{3b}$$

Note that the exponential family of constraints (3a) can be rewritten as

$$\max_{\pi\in\mathcal{P}} \sum_{u\in U} p_{uv}(\pi)(w_{uv} - \alpha_u) \leq \beta_v, \qquad \forall v \in V \tag{4}$$

which is equivalent to, for every online type $v \in V$, solving the probing problem for a star graph consisting of a single online vertex of type v and adjusted edge weights $w'_{uv} = w_{uv} - \alpha_u$.

Recall that if for every online type $v \in V$, we have a black box which can solve the maximization problem in (4) to optimality, then we have a separation oracle for the dual LP which can either establish dual feasibility or find a violating constraint given by v and π. By the equivalence of separation and optimization, this allows us to solve both the dual LP (2) and in turn the primal LP (3) using the ellipsoid method, as formalized in Proposition 1.

Proposition 1. *Given black box star graph algorithms for every online vertex type which can verify* (4) *in polynomial time, the exponential-sized LP* (2) *can be solved in polynomial time.*

An explicit statement which directly establishes Proposition 1 can be found in chapter 14 of Schrijver (1998). We should note that the ellipsoid method is only used to establish poly-time solvability in theory, and that a column generation method which adds primal variables $x_v(\pi)$ as needed on the fly is usually more efficient in practice.

In either case, while the dynamic programming approach of Purohit et al. (2019) gives us an optimal algorithm for verifying (4) in polynomial time, for the setting of general stochastic patience described in Section 5.2, the algorithm we present is only a 1/2-approximation for the star graph problem. Nonetheless, one can still use the structure of the dual LP, along with this approximate separation, to compute a $(1/2 - \epsilon)$-approximate solution to the primal LP. This is formalized in Proposition 2.

Proposition 2. *Suppose for every type $v \in V$, we are given a black box which is a κ-approximation algorithm for the maximization problem in* (4). *Then a solution to LP* (2) *with objective value at least $(\kappa - \epsilon)\mathsf{OPT}_{\mathsf{LPP}}(G)$ can be computed, in time polynomial in the problem parameters and $1/\epsilon$.*

Proposition 2 follows from the *potential-based framework* of Cheung and Simchi-Levi (2016), which is elaborated on in Online Appendix C. Armed with Propositions 1 and 2, we can use any optimal or approximately optimal algorithm for star graphs to obtain an (approximately) optimal solution to LP (2), which we can then use to solve the problem of online matching under prophet arrivals.

4.2.2. Algorithm and Analysis Based on Exponential-Sized LP. We now show how to use the LP (2) to design a $\kappa/2$-competitive online algorithm, given a feasible LP solution $x^*_v(\pi)$ which is at least $\kappa \cdot \mathsf{OPT}_{\mathsf{LPP}}(G)$, for some $\kappa \leq 1$. For each vertex $u \in U$, let $w^*_u = \sum_{v\in V} w_{uv} \sum_{\pi\in\mathcal{P}} p_{uv}(\pi)x^*_v(\pi)$ denote the expected reward of matching u according to the assignment x^*. Notice that the objective value of the given solution is $\sum_{u\in U} w^*_u$, which is at least $\kappa \cdot \mathsf{OPT}_{\mathsf{LPP}}(G)$. Using this notation, our algorithm is given in Algorithm 2. By LP constraint (2b), we have $\sum_{\pi\in\mathcal{P}} x^*_v(\pi)/q_v = 1$, so the probability distribution over policies defined in Algorithm 2 is proper. We also note that because at most polynomially many variables $x^*_v(\pi)$ will be nonzero, this distribution has polynomial-sized support and can be sampled from in polynomial time. The algorithm itself is fairly simple, selecting a policy π at random with probability proportional to LP solution $x^*_v(\pi)$ upon the arrival of a vertex v_t of type v. Then, policy π is followed in order to attempt to match the online vertex, but with two quirks: First, we skip probing any edge (π_i, v_t) for which the weight of the edge is too small (specifically, if $w_{\pi_i,v} < w^*_{\pi_i}/2$), and second, if π tells us to probe an edge (π_i, v_t) for which offline vertex π_i is unavailable, we "simulate" the probing instead, terminating with no reward if the simulated probe is successful. This simulation technique was used in Brubach et al. (2017) and is also used by our star graph algorithm in Section 5.2.

Algorithm 2 (Random Arrivals from Known Distributions (Prophet Arrivals))

```
Function OnlineMatch($U$, $V$, $\mathbf{p}$, $\mathbf{w}$):
    for $t = 1$ to $T$ do
        Online vertex $v_t$, of type $v \in V$, arrives
        Choose a policy $\pi = (\pi_1, \pi_2, \ldots, \pi_\ell)$ with probability $x^*_v(\pi)/q_v$
        for $i := 1$ to $\ell$ do
            if $w_{\pi_i,v} < w^*_{\pi_i}/2$ then
                Skip to next $i$
            else if $\pi_i$ is unmatched then
                Probe edge $(\pi_i, v_t)$ and match if successful for reward $w_{\pi_i,v}$
            else
                Simulate probing $(\pi_i, v_t)$. If successful, move to next arrival without matching $v_t$.
```

Theorem 2. *Under general edge weights and known arrival distributions, Algorithm 2 is $\kappa/2$-competitive for the online bipartite matching with patience problem, assuming we are given a solution to LP (2) with objective value at least $\kappa \cdot \mathsf{OPT}_{\mathsf{LPP}}(G)$.*

The proof of Theorem 2 is deferred to Online Appendix A. In the case of deterministic, known patience, the algorithm of Purohit et al. (2019) can be used as the black box star graph algorithms to allow verifying (4), and thus solving LP (2) exactly. In this case, $\kappa = 1$ in Theorem 2, and we have a $1/2$-competitive algorithm for online matching with patience, under prophet arrivals and arbitrary edge weights. In Section 5, we extend the classical online stochastic matching problem to consider stochastic patience and give star graph algorithms with constant-factor approximation guarantees, allowing us to apply Theorem 2 to these settings as well.

4.3. Improvement for IID Arrivals

In the case of IID arrivals, that is, where $q_{1v} = q_{2v} = \cdots = q_{Tv}$ for all vertex types $v \in V$, a slight modification to Algorithm 2 yields a competitive ratio of $(1 - 1/e)\kappa$ when given a feasible solution to LP (2) that is at least $\kappa \cdot \mathsf{OPT}_{\mathsf{LPP}}(G)$ of optimal.

The only change to the algorithm from the previous non-IID setting is that for IID arrivals, we do not skip probing vertices $u \in U$ when $w_{uv} < w^*_u/2$. The full pseudocode is given in Algorithm 3.

Algorithm 3 (Random Arrivals from Known Identical Distributions (IID Arrivals))

```
Function OnlineMatch($U$, $V$, $\mathbf{p}$, $\mathbf{w}$):
    for $t = 1$ to $T$ do
        Online vertex $v_t$, of type $v \in V$, arrives
        Choose a policy $\pi = (\pi_1, \pi_2, \ldots, \pi_\ell)$ with probability $x^*_v(\pi)/q_v$
        for $i := 1$ to $\ell$ do
            if $\pi_i$ is unmatched then
                Probe edge $(\pi_i, v_t)$ and match if successful for reward $w_{\pi_i,v_t}$
            else
                Simulate probing $(\pi_i, v_t)$. If successful, move to next arrival without matching $v_t$.
```

Theorem 3. *Under general edge weights and known IID arrivals, Algorithm 3 is $\kappa(1 - 1/\mathrm{e})$-competitive for the online bipartite matching with patience problem, assuming we are given a solution to LP (2) with objective value at least $\kappa \cdot \mathsf{OPT}_{\mathsf{LPP}}(G)$.*

The proof of Theorem 3 is deferred to Online Appendix A. As with Theorem 2, the dynamic program of Purohit et al. (2019) can be used to get an optimal LP solution, so that Theorem 3 gives a competitiveness of $1 - 1/\mathrm{e}$ for Algorithm 3 under the classical deterministic patience setting. Section 5 extends the result to online stochastic matching with stochastic patience, by providing black box algorithms for those settings which can then be used to find approximately optimal solutions to LP (2) as per Proposition 2.

4.4. Improvement for Vertex Weights

With a new analysis, we can show that Algorithm 3 still achieves a competitive ratio of $1 - 1/\mathrm{e}$ in the prophet (nonstationary) setting in the case of *vertex weights*.

Theorem 4. *Under vertex weights and known arrival distributions, Algorithm 3 is $\kappa(1 - 1/e)$-competitive for the online bipartite matching with patience problem, assuming we are given a solution to LP (2) with objective value at least $\kappa \cdot \mathsf{OPT}_{\mathsf{LPP}}(G)$.*

The proof of Theorem 4 is deferred to Online Appendix A. It has identical implications as Theorem 3, with the improvement beyond the $1/2$-guarantee of Theorem 2 coming from the vertex-weighted assumption instead of the IID assumption.

5. Algorithms for Star Graphs

All of the algorithms in Section 4 make use of a *black box algorithm* for solving the special case of a single customer (called the "star graph" problem, since it corresponds to a star graph with the customer as the center vertex): Under adversarial arrivals, the black box is used for each arriving vertex, while for prophet arrivals it is used as a separation oracle for the dual LP. For the classic problem of online matching with stochastic edges and finite patience, we may use the dynamic program of Purohit et al. (2019) as this black box. Since this dynamic program is optimal for star graphs, $\kappa = 1$ in all of the results of Section 4. In this section, we consider the problem where the patience of the customer is not known but is instead determined by some stochastic process. These algorithms can be used as black boxes for the algorithms in Section 4, giving competitive ratios for online matching with stochastic edges (with multiple customers) under these new stochastic patience settings. Finally, we also give an algorithm for an alternate setting where the customer has a deterministic patience, but items arrive over time according to Bernoulli processes. Throughout, we use m to denote the number of items, as in Section 3.

5.1. Constant Hazard Rate

In this setting, the patience is random and unknown, with a constant "hazard rate" r_i for each $i \in [m]$.[8] When an attempted match with u_i is unsuccessful, the customer v runs out of patience with probability r_i and remains available for another match attempt with probability $1 - r_i$. When the hazard rate r_i is the same for all $i \in [m]$ (i.e., $r_i = r$ for some r and every i), this is equivalent to the patience having a constant hazard rate of r.

Theorem 5. *For maximizing the expected weight of the matched item in the (item-dependent) Constant Hazard Rate patience model, it is optimal to probe items in decreasing order of*

$$\frac{w_i p_i}{p_i + (1 - p_i) r_i}.$$

Theorem 5 tells us that, in the special case where the patience distribution can be modeled with individual per-item hazard rates, we can achieve an optimal star graph probing strategy. As such, we can apply all of the algorithms and results from Section 4 with $\kappa = 1$.

5.2. Arbitrary Patience Distributions

We address the case where the patience is stochastic (and unknown) and can follow an arbitrary known distribution. Without loss of generality, we can assume the patience distribution has finite support (more specifically, that the patience $\theta \in [m]$); this is because a patience greater than m is equivalent to a patience of m (additionally, a patience of zero indicates a vertex which can never be matched by any algorithm, since no probes can be made, so we can simply ignore such vertices). We use an LP-based approach for this problem. We denote by q_θ the probability that the patience of the online vertex v is *at least* θ. Notice that $1 = q_1 \geq q_2 \geq \cdots \geq q_n$. Our approach utilizes the LP (5) described in the next paragraph. The variables are $x_{j\theta}$, which correspond to the probability of *attempting to match* with j on the θth attempt. The value s_θ represents the probability that the online vertex is available for a θth match attempt, meaning its patience is at least θ and all previous match attempts were unsuccessful. This can be calculated from the $x_{j\theta}$, q_θ, and p_j values.

$$\max \sum_{j=1}^{m} w_j p_j \sum_{\theta=1}^{m} x_{j\theta} \tag{5}$$

$$\text{subject to} \sum_{\theta'=\theta}^{m} x_{j\theta'} \leq s_\theta, \qquad \forall j \in \{1,2,\ldots,m\}, \theta \in \{1,2,\ldots,m\} \tag{5a}$$

$$\sum_{j=1}^{m} x_{j\theta} \leq s_\theta, \qquad \forall \theta \in \{1,2,\ldots,m\} \tag{5b}$$

$$x_{j\theta} \geq 0, \qquad \forall j \in \{1,2,\ldots,m\}, \theta \in \{1,2,\ldots,m\}$$

$$s_1 = 1, \tag{5c}$$

$$s_\theta = \frac{q_\theta}{q_{\theta-1}} \left(s_{\theta-1} - \sum_{j=1}^{m} p_j x_{j,\theta-1} \right) \qquad \forall \theta \in \{2,\ldots,m\} \tag{5d}$$

As an example, consider the e-commerce application, where we wish to offer items to a customer one by one. The variable $x_{j\theta}$ indicates the probability that the customer is offered item j after $\theta - 1$ other items have already been offered (and rejected). This can only happen if the customer's patience is at least θ *and* the customer has not already purchased an item (prior to the θth offer), since otherwise no offers can be made at this point. The value s_θ denotes precisely this probability, that is, the probability that the customer's patience is at least θ and the customer has rejected all previous offers. This suggests a simple constraint: $x_{j\theta} \leq s_\theta$. However, two stronger conditions can be given: first, no valid strategy can offer item j at any point on or after the θth attempt, if the customer has patience less than θ or purchases an item offered before the θth. Summing over all offers after $\theta - 1$, $\sum_{\theta'=\theta} x_{j\theta'}$ is the probability that item j is offered at or after the θth step, and the reasoning above gives us the constraints (5a). Further, if the customer is unavailable (due to having purchased an item or running out of patience) for the θth attempt, then no item can be offered on the θth attempt; thus, the probability of offering an item on attempt number θ cannot exceed the probability that the customer is available for the θth attempt. This gives constraints (5b). We note that the family of constraints (5a) differs from similar time-indexed LPs in the literature (Ma 2018), in that there is a constraint for the sum starting at every attempt θ instead of a single constraint where $\theta = 1$.

Constraints (5c) and (5d) simply give a closed-form expression for the quantities s_θ, where $\frac{q_\theta}{q_{\theta-1}}$ is understood to be zero if both q_θ and $q_{\theta-1}$ are zero. Finally, since $x_{j\theta}$ corresponds to a probability, we require $x_{j\theta} \in [0, 1]$ (we do not explicitly write the constraint $x_{j\theta} \le 1$ in the LP above, since it is redundant, being implied by (5b) for $\theta = 1$, since $s_1 = 1$). We can see that this LP upper bounds the optimal algorithm, since taking $x_{j\theta}$ to be the probability of the algorithm probing j on attempt θ for all j and θ, we get a feasible solution to the LP with objective value equal to the algorithm's expected weight matched.

Our algorithm is simple: we solve LP (5) to get an optimal solution x^*, along with the values s^*. Then, when making the θ^{th} probe, choose each offline vertex $j = 1, \dots, n$ with probability $x^*_{j\theta}/s^*_\theta$ (note that if $s^*_\theta = 0$ then $x^*_{j\theta} = 0$), which defines a proper probability distribution by (5b). If a vertex j is chosen to be probed, but has already been unsuccessfully probed in a previous attempt, we "simulate" probing j instead, and *terminate* with no reward if the simulated probe is successful. This simulation technique is important in ensuring that the probability of surviving to the θ^{th} attempt is consistent with the LP value s^*_θ.

Example 1. We provide an example to illustrate our LP and algorithm. Let $m = 2$, and suppose there are two items with weights $w_1 = 1, w_2 = 2$ and probabilities $p_1 = 3/4, p_2 = 1/4$.

First, suppose $q_1 = q_2 = 1$, that is, the patience is deterministically two. Then the nonzero x-values in the optimal LP solution are $x_{2,1} = 1, x_{1,2} = 0.75$. This corresponds to probing item 2 (with the higher reward if it succeeds) first, and if the probe fails (occurring w.p. 3/4), probing item 1 afterward. The expected reward is $w_2p_2 + (1 - p_2)w_1p_1 = 2/4 + 3/4 \cdot 3/4 = 17/16 = 1.0625$, which is also the optimal objective value of the LP.

Now consider a more interesting example where $q_1 = 1, q_2 = 1/3$. Then the nonzero x-values in the optimal LP solution are $x_{1,1} = 0.9, x_{1,2} = 0.1, x_{2,1} = 0.1$. Our algorithm in this case will probe item 1 first w.p. 0.9, and otherwise (w.p. 0.1) probe item 2 first. If it survives to the second probe, which occurs w.p. $s_2 = 0.1$, it will always probe item 1. However, note that this will be a "simulated" probe (which generates no reward) unless item 2 was probed on the first probe, making item 1 still available for the second probe. The probability of this occurring is $0.1 \cdot 3/4 \cdot 1/3 = 0.025$. Therefore, the expected reward of our algorithm is $(0.9 + 0.025) \cdot 3/4 + 0.1 \cdot 2/4 \approx 0.743$. Meanwhile, the LP optimal value is 0.8.

We note that in this case, the best algorithm is to probe item 1 first (which is the "safe bet," given that the customer has a 2/3 chance of departing after the first probe) followed by item 2, which would have expected reward $19/24 \approx 0.791$, worse than the LP value. It would have been better than our algorithm, though. However, note that the optimal ordering given many items and an arbitrary patience distribution is generally nontrivial to solve (even in our examples, the optimal ordering switched from 2,1 to 1,2 depending on the patience distribution), and to our knowledge the best approximation algorithm is the 1/2-approximation provided by our randomized algorithm.

Theorem 6. *The online algorithm based on LP (5) is a 1/2-approximation for the star graph probing problem, for an arbitrary patience distribution which is given explicitly.*

The proof of Theorem 6 is in Online Appendix A; we note that the analysis in the proof is tight.

We further note that the result of Theorem 6 compares to a benchmark (LP (5)) that does *not* know the full realization of the patience values in advance. This is necessary, since Theorem 10 states that comparing to a benchmark which knows the patience in advance leads to arbitrarily bad competitive ratios.

Theorem 6 allows for us to solve online matching problems when the patience of each customer is stochastic and follows an *arbitrary* distribution that is known to the algorithm. We simply use our star graph algorithm as a black box for our algorithms in Section 4, with $\kappa = 1/2$. This gives us $\frac{1}{4}$-competitive algorithms for vertex-weighted adversarial arrivals and edge-weighted prophet arrivals; as per Theorems 3 and 4, we have improved competitive ratios of $\frac{1}{2}(1 - 1/\mathrm{e})$ under known IID arrivals (even with arbitrary edge weights) and vertex-weighted prophet arrivals (even when the distributions are not identical).

5.3. Item Arrivals

Next, we consider a different setting in which after a customer arrives, the "items" (interpreted as contractors in an online platform) are initially unavailable and only show up (to acknowledge that they can do the customer's job) following Bernoulli processes. More specifically, each item $i \in [m]$ has two given probabilities: the matching probability p_i and an arrival probability q_i. The customer has a known deterministic patience θ, and the process unfolds in discrete time steps; at time $t \in \{1, \dots, \theta\}$, the algorithm must choose at most one item that has arrived to offer to the customer. After time $t = \theta$, if no item has been purchased, the customer runs out of patience and becomes permanently unavailable for matches. In contrast to the other patience settings, in this setting the patience corresponds to the *amount of time* the customer is willing to wait, rather than the *number of items* they may be offered. Thus, if the algorithm makes no offer at time t (because it is waiting for items to become available), we still move one step closer to exhausting the customer's patience.

When the customer first arrives (immediately prior to time $t = 1$), no items are available to be offered. However, at each time step, each item i becomes available independently with probability q_i. Once an item i becomes available to the customer, say initially at time t, then it can be offered to the customer at most once, at any time step $t' \geq t$. When item i is offered, the customer purchases it with probability p_i, in which case a weight of w_i is achieved and the process terminates; with probability $1 - p_i$, the item is not purchased and the process immediately proceeds to the next time step. As with all star graph problems, our goal is to develop an algorithm which maximizes the expected weight of the item sold to the customer (achieving a weight of zero if no item is sold).

We begin with a linear programming relaxation of the problem.

$$\text{LP} := \max \sum_{i=1}^{m} w_i x_i p_i \tag{6}$$

$$\text{subject to} \sum_{i=1}^{m} x_i p_i \leq 1 \tag{6a}$$

$$\sum_{i=1}^{m} x_i \leq \theta \tag{6b}$$

$$x_i \leq 1 - (1 - q_i)^{\theta} \quad \forall i \in \{1, 2, \ldots, m\} \tag{6c}$$

$$x_i \geq 0 \quad \forall i \in \{1, 2, \ldots, m\} \tag{6d}$$

We call an item "large" if $q_i \geq c/\theta$. Otherwise, if $q_i < c/\theta$, we say that item u_i is "small." Let $I_{\text{LARGE}} = \{i \in [m] | x_i \geq c/\theta\}$ denote the set of large items, and $I_{\text{SMALL}} = \{i \in [m] | x_i < c/\theta\}$ denote the set of small items. Our algorithm first solves the LP (6) to obtain an optimal solution $(x_i)_{i \in [m]}$; then, it makes use of one of two different strategies, choosing between the two depending on relative contribution of large versus small items to the LP objective. The motivation here is as follows: Intuitively, we wish to choose to offer an item i with some probability proportional to x_i. However, if most of the contribution to the objective value in the LP comes from small items, there may be a high probability of no items arriving in any one time step; in this case, we may be better off simply offering any item that arrives in a time step where we are lucky enough to have an arrival.

5.3.1. The LARGE Strategy.

First, let $\rho \in \left(0, \frac{1}{2}\right)$ be a fixed parameter. Our strategy π_{LARGE} does the following: at each time step, we select an item at random. When selecting a random item, we choose item i with probability x_i/θ. With probability $1 - \sum_{i=1}^{m} x_i/\theta$, we select no item. It follows from constraint (6b) that this forms a valid distribution.

The algorithm selects this item at random, and if the item has arrived and has not yet been offered, we offer it to the buyer with probability ρ (and with probability $1 - \rho$ we make no offer).

5.3.2. The SMALL Strategy.

Our strategy π_{SMALL} does the following: At each time step, if at least one small item arrives in that step, choose one of the small arrivals at random and offer it to the buyer. We ignore large items. Any small item which arrives and is not chosen is permanently discarded (i.e., it will never be offered to the buyer).

5.3.3. The Full Algorithm.

We fix a parameter $\varphi \in (0, 1)$. First, if $\theta = 1$, we simply take the optimal choice, offering the item with the highest expected reward among items that arrived. Then, for $\theta \geq 2$, if $\sum_{i \in I_{\text{LARGE}}} w_i x_i p_i \geq (1 - \varphi)\text{LP}$, we use strategy π_{LARGE} at every time step. Otherwise, $\sum_{i \in I_{\text{SMALL}}} w_i x_i p_i > \varphi\text{LP}$, and we use π_{SMALL} at every time step.

We show that this algorithm is a 0.027-approximation for the problem. This is done by considering the two cases (corresponding to using the LARGE and SMALL strategies) separately.

Lemma 3. *If $\sum_{i \in I_{\text{LARGE}}} w_i x_i p_i \geq (1 - \varphi)\text{LP}$, then π_{LARGE} achieves an expected matching weight of $(1 - \varphi)c\rho(1 - 2\rho)\text{LP}$.*

Lemma 4. *If $\sum_{i \in I_{\text{LARGE}}} w_i x_i p_i < (1 - \varphi)\text{LP}$, then π_{SMALL} achieves an expected matching weight of at least*

$$\varphi\left(1 - \frac{c}{2}\right)^{\frac{2}{1-(1-c/2)^2}}\left(\frac{1}{c} - \frac{e^{-c}}{1 - e^{-c}}\right)\text{LP} \tag{7}$$

Using the bounds for both the LARGE and SMALL strategies, we can now give our final result.

Theorem 7. *For an appropriate choice of parameters φ, ρ, c, our algorithm is a 0.027-approximation.*

Lemmas 3 and 4 and Theorem 7 are proved in Online Appendix A. Our result for this setting can be used as a black box for a new kind of online stochastic matching problem with two-sided arrivals, where after the arrival of each customer, all items (contractors in an online labor platform) are initially unavailable, and arrive over time following Bernoulli processes (when they "discover" the customer's task). These acknowledgments "reset" after each customer, who presents a new job, and we note that that each (contractor, customer)-pair can have a different rate for its Bernoulli process of the contractor arriving, as well as a different probability for the customer accepting that contractor. A contractor, once matched, spends the time horizon (e.g., one day) doing that task and hence never returns. For such an online matching problem, we may use our strategy as a black box for the algorithms of Section 4, where we have $\kappa = 0.027$, to get a constant-factor competitive ratio for any finite market size and time horizon.

6. Negative Results

6.1. Stochasticity Gap

The *stochasticity gap* is a fundamental gap in linear programming relaxations for stochastic problems which replace probabilities with deterministic fractional weights.

The notion was first discussed informally in Brubach et al. (2017) and was later also observed by Purohit et al. (2019) (where they referred to it as a "probing gap"). When these LP relaxations are used as upper bounds on the offline optimal solution, or as a benchmark for the competitive ratio, the stochasticity gap represents a barrier to the best achievable competitive ratio. One interpretation of such a result is that better competitive ratios are not possible. However, one may alternatively view it as a result showing the limitations of using a particular LP as a benchmark for competitive ratios.

We present a stochasticity gap for a common LP relaxation of the online matching problem with stochastic rewards. Recall from Section 4.1.1 that LP (1) with the constraints (1a)–(1c) (and excluding our additional family of constraints (1d)) is a standard LP relaxation for bipartite matching with (known) patience constraints and adversarial arrivals. This is essentially an extension of the "Budgeted Allocation" LP from Mehta and Panigrahi (2012) to include the patience constraints. For convenience, we reproduce this standard LP below in the vertex-weighted setting.

$$\begin{aligned}
\max \quad & \sum_{u \in U} \sum_{v \in V} x_{u,v} p_{u,v} w_u \\
\text{subject to} \quad & \sum_{v \in V} x_{u,v} p_{u,v} \le 1 && \forall u \in U \\
& \sum_{u \in U} x_{u,v} p_{u,v} \le 1 && \forall v \in V \\
& \sum_{u \in U} x_{u,v} \le \mathbb{E}[\theta_v] && \forall v \in V \\
& 0 \le x_{u,v} \le 1 && \forall u \in U, v \in V
\end{aligned}$$

Simple LP formulations like this, while useful, can give too large of an upper bound on the performance of any offline algorithm and thus make it difficult to get larger competitive ratios. As such, more complex (and, often, exponentially sized) LPs have been used in recent work (see, e.g., Gamlath et al. 2019) to achieve better results. Our LP-based techniques in Section 4 use different exponential-sized LPs to overcome the limitations of stochasticity gaps.

We start with a simple example demonstrating the notion of a stochasticity gap, where the bipartite graph has a single offline vertex u, and n online vertices arriving in any order. Suppose $p_{uv} = 1/n$ and $w_{uv} = 1$ for all online vertices $v \in V$. The LP given by (1a)–(1c) can assign $x_{uv} = 1$ for all edges, achieving an objective value of one. However, the best any online algorithm can do is probe the single edge (u, v) whenever vertex v arrives online, which matches the single offline vertex u with probability $1 - 1/e$. Thus, it is impossible for any online algorithm to guarantee a matching of expected weight better than $(1 - 1/e)$ times the LP value. This establishes a stochasticity gap of $1 - 1/e$ for this formulation and suggests that if we wish to beat the $1 - 1/e$ barrier, we must use a different LP benchmark. However, the stochasticity gap for the LP of (1a)–(1c) is even worse. To establish this, we consider a complete bipartite graph with n vertices on each side, and edge probabilities $1/n$; a result on random graphs then implies Theorem 8, whose proof is in Online Appendix A.

Theorem 8. *The LP given by the objective function (1) and constraints (1a)–(1c) has a stochasticity gap of at most* ≈ 0.544.

We should note that Fata et al. (2019) establish a smaller upper bound of $1 - \ln(2 - 1/e) \approx 0.51$ relative to this LP, but they restrict to online probing algorithms. Our higher upper bound holds even for the offline optimal matching, hence reflecting a true "stochasticity gap."

6.2. The 0.5 Upper Bound for SimpleGreedy

As defined in Mehta and Panigrahi (2012), an *opportunistic* algorithm for the *Stochastic Rewards* setting is one which always attempts to probe an edge incident to an online arriving vertex $v \in V$ if one exists. The work of Mehta and Panigrahi (2012) showed that in the unweighted *Stochastic Rewards* ($\theta_v = 1$ for all online vertices $v \in V$) problem, any opportunistic algorithm achieves a competitive ratio of $1/2$. The simplest opportunistic algorithm is the one which, when $v \in V$ arrives online, chooses a neighbor $u \in U$ of v arbitrarily and probes the edge (u, v). We call this algorithm "SimpleGreedy." Since SimpleGreedy is opportunistic, the result of Mehta and Panigrahi (2012) shows that SimpleGreedy achieves a competitive ratio of at least $1/2$; Theorem 9 shows that this is tight even when restricted to small, uniform p.

Theorem 9. *There exists a family of unweighted graphs under stochastic rewards and adversarial arrivals for which SimpleGreedy achieves a competitive ratio of at most 1/2 even when all edges have uniform probability* $p = O(1/n)$.

We present our construction here. Let k be a fixed positive integer constant. Let $U = U_0 \cup U_n$, where U_0 and $U_n = \{u_1, \ldots, u_n\}$ are disjoint, and $|U_0| = k$. Let $V = V_0 \cup V_n$ where V_0 and $V_n = \{v_1, \ldots, v_n\}$ are disjoint, and $|V_0| = kn^2$. Let $E = E_0 \cup E_n$ where $E_0 = U_0 \times V$ and $E_n = \{(u_i, v_i) | i = 1, \ldots, n\}$. Let $p = k/n$.

For the bipartite graph $G(U, V; E)$, an offline algorithm can achieve a matching of expected size at least $2k$ by first probing edges $(u, v) \in U_0 \times V_0$ until all edges are probed or the maximum possible successful matches, k, is achieved. This strategy achieves k successful matches among these edges in expectation. Then, the offline optimal will probe all edges of E_n in any order, achieving an expected number of successful matches of k. The total expected size of the achieved matching is then $2k$. We complete the proof of Theorem 9 by showing that an online algorithm cannot earn more than $k + o(k)$, in Online Appendix A.

6.3. Hardness of Unknown Patience

We now show that when offering items to a single customer with random patience, one should not be comparing to a benchmark that knows the realization of the patience in advance, or else the competitive ratio will be zero. The same counterexample shows for single-item IID-valued online accept/reject problems that the competitive ratio will be zero if the number of arrivals is unknown, recovering the result of Alijani et al. (2020).

Theorem 10. *For the star graph probing problem with patience θ drawn from an arbitrary distribution,*[9] *the reward of an online algorithm relative to a clairvoyant who sees the realization of θ in advance must be zero, even if there are infinite copies of every offline vertex.*

Theorem 10 is proved in Online Appendix A, and its construction is presented below. The significance of allowing infinite copies of offline vertices is the following. Essentially, we are left with a pricing problem where there are an unknown θ number of opportunities to make a single sale to a customer; the different offline vertices' weights correspond to different prices that can be tried, and after each trial we get an independent realization (because of the infinite copies) whose probability depends on the price. One can further transform such an instance into an online accept/reject problem facing a stream of θ IID draws, where the pricing decisions correspond to acceptance thresholds. Therefore, our hardness result implies the following. Although already known to Alijani et al. (2020, appendix A.1), we rederive it using our construction to articulate the connection, which we believe is instructive.

Corollary 1. *Consider the simple optimal stopping problem where an online algorithm can accept at most one of θ values that are drawn IID from a known distribution and presented one by one. If θ is unknown, then the competitive ratio is zero.*

6.3.1. Our Construction.

Fix a positive integer k and let m be another positive integer that we will drive to ∞. Consider a star graph, that is, a bipartite graph with many offline vertices $u \in U$ and a single online vertex v. Consider the following distribution over the patience of v:

$$\tilde{\theta}_v = \begin{cases} m^{2i} & w.p.\ m^{-i} - m^{-i-1} \qquad \forall i = 0, \dots, k-1; \\ m^{2k} & w.p.\ m^{-k}. \end{cases} \tag{8}$$

In our construction, there are m^{2k} identical offline vertices for each $i = 0, \dots, k$, with weight m^i and probability m^{-2i}. We note that m^{2k} is greater than the largest possible realization of $\tilde{\theta}_v$, so the constraints on the availability of offline vertices are never binding. That is, *our construction applies even in the more restrictive setting* where there are infinite copies of every offline vertex.

6.3.2. Intuition Behind Hardness.

In our construction, there are essentially an unknown number of opportunities to sell a single item. During each opportunity, one must choose a consumption option $i = 0, \dots, k$, which has an m^{-2i} probability of selling the item at price m^i. The immediate reward from consumption option i is $m^i \cdot m^{-2i} = m^{-i}$, which is decreasing in i, but smaller indices of i also have a higher chance of closing the sale and eliminating future opportunities. Therefore, there is a trade-off between offering "longshot" prices with a high index of i (desirable if a large number of opportunities remain) and the "safe" option $i = 0$ which makes a sale w.p. $m^{-0} = 1$ (desirable on the final opportunity). Our proof of Theorem 10 in Online Appendix A shows that a clairvoyant who tries only option i when they know the patience will be m^{2i}, for all $i = 0, \dots, k$, can earn $\approx (1 - 1/e)(k+1)$. Meanwhile, any online algorithm is best off using the "safe" option $i = 0$ on the first try and finishing, since the customer only has a small chance of having patience greater than one (we prove this through backward induction on the optimal dynamic program). This establishes an unbounded separation when k is taken to be large in our construction.

6.3.3. Transformed Hard Instance to Establish Corollary 1.

For concreteness, we show how to transform our construction to the accept/reject problem. The IID draws from the distribution should take one of $k + 1$ possible values, indexed by $i = 0, \dots, k$. The decision each period is to set a threshold on the minimum acceptable value, where each option $i = 0, \dots, k$ should correspond to a "consumption option" that has probability m^{-2i} of accepting. Therefore, the probability of the IID draw taking value index i for all $i = 0, \dots, k-1$ should be $m^{-2i} - m^{-2(i+1)}$ and the probability of value index $i = k$ should be m^{-2k}, so that accepting all levels with index at least i has probability

$$(m^{-2i} - m^{-2(i+1)}) + (m^{-2(i+1)} - m^{-2(i+2)}) + \cdots + m^{-2k} = m^{-2i}$$

of making an acceptance. Now, the exact values for each index i must be calibrated so that the immediate reward from each consumption option i is m^{-i}. Using backward induction over $i = k, \dots, 1$, we can solve that the value for index $i = k$ should be m^k and that the value for each index $i = 0, \dots, k-1$ should be

$$\frac{m^{-i} - m^{-(i+1)}}{m^{-2i} - m^{-2(i+1)}} = \frac{m^{i+2} - m^{i+1}}{m^2 - 1} = \frac{m^{i+1}}{m+1}.$$

This completes the construction of our transformed instance for the accept/reject problem.

6.3.4. Applying Yao's Minimax Principle.

Finally, we explain why in Corollary 1, there is no difference between θ being completely unknown and θ being drawn from a known distribution (but competing against a clairvoyant who knows its realization in advance). Formally, for any patience θ and any (deterministic) nonclairvoyant algorithm ψ, let $\mathsf{ALG}(\psi, \theta)$ denote the algorithm's expected reward when the patience realizes to θ. Meanwhile, let $\mathsf{OPT}(\theta)$ denote the clairvoyant's expected reward when the patience is known to be θ. Let D denote a distribution over patiences θ, and Ψ denote a distribution over algorithms ψ. Yao's minimax principle says that

$$\sup_{\Psi} \inf_{\theta} \frac{\mathbb{E}_{\psi \sim \Psi}[\mathsf{ALG}(\psi, \theta)]}{\mathsf{OPT}(\theta)} = \inf_{D} \sup_{\psi} \mathbb{E}_{\theta \sim D}\left[\frac{\mathsf{ALG}(\psi, \theta)}{\mathsf{OPT}(\theta)}\right]$$
$$= \inf_{D} \frac{\sup_{\psi} \mathbb{E}_{\theta \sim D}[\mathsf{ALG}(\psi, \theta)]}{\mathbb{E}_{\theta \sim D}[\mathsf{OPT}(\theta)]}$$

(where the second equality holds via rescaling worst-case distributions D by $\mathsf{OPT}(\theta)$). The existence of our family of distributions in (8) shows that the right-hand side expression, and hence all of these expressions, equals zero. The left-hand side expression equaling zero implies that for any fixed (randomized) online algorithm that does not know the value of θ in advance, an adversary can always set a horizon length θ for which the algorithm performs unboundedly worse relative to $\mathsf{OPT}(\theta)$.

Acknowledgments
Will Ma thanks Ningyuan Chen, Jake Feldman, Calum MacRury, and Danny Segev for instructive discussions relevant to this paper and Kangning Wang for pointing the authors to the reference Alijani et al. (2020). A preliminary version of this paper appeared in AISTATS (Brubach et al. 2021).

Endnotes

[1] This includes everything that is known about the customer at the time of their arrival, including purchase probabilities, patience distribution, edge weights, etc.

[2] Similar ideas have appeared in Cheung et al. (2022), who consider abstract "actions" that have different immediate rewards and different consumption distributions over resources. However, their focus is on learning these distributions.

[3] However, we acknowledge that their $1/\mathrm{e}$-approximation holds against a stronger benchmark that knows the patience in advance. This is only possible under some special cases of the patience distribution: as we show in Section 6.3, such a result is *impossible* for the general patience distributions we consider, so our LP relaxation (necessarily) does not know the patience in advance.

[4] Their paper focuses on the *offline* matching with stochastic edges problem, which we do not consider in this literature review.

[5] The techniques in Brubach et al. (2017) also involved solving a star graph problem with a black box. However, that work first solved an LP for a bipartite graph, and then used a black box probing algorithm to essentially round and probe the LP solution on the induced star graphs of arriving vertices. This differs from our work, which uses algorithms for stochastic matching on star graphs as black boxes to solve a more sophisticated LP and then uses that LP solution to guide the online algorithm.

[6] This is because it was the arrival model of original focus in prophet inequality papers (Krengel and Sucheston 1977).

[7] We note that Borodin et al. (2022) derive results against a stronger benchmark, which can switch back and forth between online vertices.

[8] Throughout this paper, we use the notation $[m] := \{1, 2, \ldots, m\}$.

[9] See Section 5.2 for the exact problem statement. There we compared with an LP that also did not know the realization of the patience in advance, the importance of which is fully justified by the present theorem.

References

Adamczyk M, Grandoni F, Mukherjee J (2015) Improved approximation algorithms for stochastic matching. Bansal N, Finocchi I, eds. *Algorithms – ESA 2015: 23rd Annual European Symposium,* Patras, Greece, September 14–16 (Springer, Berlin), 1–12.

Aggarwal G, Goel G, Karande C, Mehta A (2011) Online vertex-weighted bipartite matching and single-bid budgeted allocations. *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms* (SIAM, Philadelphia), 1253–1264.

Alaei S, Hajiaghayi M, Liaghat V (2012) Online prophet-inequality matching with applications to ad allocation. *Proceedings of the 13th ACM Conference on Electronic Commerce* (Association for Computing Machinery, New York), 18–35.

Alijani R, Banerjee S, Gollapudi S, Munagala K, Wang K (2020) Predict and match: Prophet inequalities with uncertain supply. *Proc. ACM Measurement Anal. Comput. Systems* 4(1):1–23.

Aouad A, Saritaç Ö (2022) Dynamic stochastic matching under limited time. *Oper. Res.* 70(4):2349–2383.

Aveklouris A, DeValve L, Ward AR (2021) Matching impatient and heterogeneous demand and supply. Preprint, submitted February 4, https://doi.org/10.48550/arXiv.2102.02710.

Bansal N, Gupta A, Li J, Mestre J, Nagarajan V, Rudra A (2010) When LP is the cure for your matching woes: Improved bounds for stochastic matchings. *Algorithms – ESA 2010: 18th Annual European Symposium, Liverpool, UK, September 6–8* (Springer, Berlin), 218–229.

Borodin A, MacRury C, Rakheja A (2021) Secretary matching meets probing with commitment. Wootters M, Sanita L, eds. *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021)* (Schloss Dagstuhl-Leibniz-Zentrum für Informatik, Wadern, Germany), 13:1–13:23.

Borodin A, MacRury C, Rakheja A (2022) Prophet matching in the probe-commit model. Chakrabarti A, Swamy C, eds. *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2022)* (Schloss Dagstuhl-Leibniz-Zentrum für Informatik, Wadern, Germany), 46:1–46:24.

Brubach B, Grammel N, Ma W, Srinivasan A (2021) Follow your star: New frameworks for online stochastic matching with known and unknown patience. *Proc. Machine Learn. Res.* 130: 2872–2880.

Brubach B, Sankararaman KA, Srinivasan A, Xu P (2017) Attenuate locally, win globally: An attenuation-based framework for online stochastic matching with timeouts. *AAMAS '17: Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems* (International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC), 1223–1231.

Brubach B, Sankararaman KA, Srinivasan A, Xu P (2020) Online stochastic matching: New algorithms and bounds. *Algorithmica* 82(10):2737–2783.

Chen N, Li A, Yang S (2021) Revenue maximization and learning in products ranking. *EC '21: Proceedings of the 22nd ACM Conference on Economics and Computation* (Association for Computing Machinery, New York), 316–317.

Cheung WC, Simchi-Levi D (2016) Efficiency and performance guarantees for choice-based network revenue management problems with flexible products. Preprint, submitted August 15, https://dx.doi.org/10.2139/ssrn.2823339.

Cheung WC, Ma W, Simchi-Levi D, Wang X (2022) Inventory balancing with online learning. *Management Sci.* 68(3):1776–1807.

Derakhshan M, Golrezaei N, Manshadi V, Mirrokni V (2022) Product ranking on online platforms. *Management Sci.* 68(6):4024–4041.

Fata E, Ma W, Simchi-Levi D (2019) Multi-stage and multi-customer assortment optimization with inventory constraints. Preprint, submitted August 26, https://dx.doi.org/10.2139/ssrn.3443109.

Feldman J, Segev D (2019) Improved approximation schemes for MNL-driven sequential assortment optimization. Preprint, submitted August 21, https://dx.doi.org/10.2139/ssrn.3440645.

Feldman J, Mehta A, Mirrokni V, Muthukrishnan S (2009) Online stochastic matching: Beating 1-1/e. *Proceedings of the 50th Annual Symposium on Foundations of Computer Science* (IEEE, New York), 117–126.

Gamlath B, Kale S, Svensson O (2019) Beating greedy for stochastic bipartite matching. *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '19)* (Society for Applied and Industrial Mathematics, Philadelphia), 2841–2854.

Golrezaei N, Nazerzadeh H, Rusmevichientong P (2014) Real-time optimization of personalized assortments. *Management Sci.* 60(6):1532–1551.

Goyal V, Udwani R (2023) Online matching with stochastic rewards: Optimal competitive ratio via path-based formulation. *Oper. Res.* 71(2):563–580.

Huang Z, Zhang Q (2020) Online primal dual meets online matching with stochastic rewards: Configuration lp to the rescue. *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing* (Association for Computing Machinery, New York), 1153–1164.

Jiang J, Ma W, Zhang J (2022) Tight guarantees for multi-unit prophet inequalities and online stochastic knapsack. *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)* (SIAM, Philadelphia), 1221–1246.

Kalyanasundaram B, Pruhs KR (2000) An optimal deterministic algorithm for online b-matching. *Theoret. Comput. Sci.* 233(1–2): 319–325.

Karp RM, Vazirani UV, Vazirani VV (1990) An optimal algorithm for on-line bipartite matching. *Proceedings of the Twenty-Second Annual ACM Symposium on Theory of Computing* (Association for Computing Machinery, New York), 352–358.

Kempe D, Mahdian M (2008) A cascade model for externalities in sponsored search. Papadimitriou C, Zhang S, eds. *International Workshop on Internet and Network Economics* (Springer, Berlin), 585–596.

Kerimov S, Ashlagi I, Gurvich I (2021) On the optimality of greedy policies in dynamic matching. Preprint, submitted September 6, https://dx.doi.org/10.2139/ssrn.3918497.

Kessel K, Shameli A, Saberi A, Wajc D (2022) The stationary prophet inequality problem. *Proceedings of the 23rd ACM Conference on Economics and Computation* (Association for Computing Machinery, New York), 243–244.

Krengel U, Sucheston L (1977) Semiamarts and finite values. *Bull. Amer. Math. Soc. (N.S.)* 83(4):745–747.

Ma W (2018) Improvements and generalizations of stochastic knapsack and Markovian bandits approximation algorithms. *Math. Oper. Res.* 43(3):789–812.

Ma W, Simchi-Levi D (2020) Algorithms for online matching, assortment, and pricing with tight weight-dependent competitive ratios. *Oper. Res.* 68(6):1787–1803.

Ma W, Simchi-Levi D, Zhao J (2021) Dynamic pricing (and assortment) under a static calendar. *Management Sci.* 67(4):2292–2313.

Mehta A (2012) Online matching and ad allocation. *Foundations Trends Theoret. Comput. Sci.* 8(4):265–368.

Mehta A, Panigrahi D (2012) Online matching with stochastic rewards. *FOCS '12: Proceedings of the 2012 IEEE Annual Symposium on Foundations of Computer Science* (IEEE, New York), 728–737.

Mehta A, Waggoner B, Zadimoghaddam M (2015) Online stochastic matching with unequal probabilities. *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms* (SIAM, Philadelphia), 1388–1404.

Purohit M, Gollapudi S, Raghavan M (2019) Hiring under uncertainty. *Proc. Machine Learn. Res.* 97:5181–5189.

Schrijver A (1998) *Theory of Linear and Integer Programming* (John Wiley & Sons, Inc., Hoboken, NJ), 1–484.

Brian Brubach is a faculty member in the Computer Science Department of Wellesley College.

Nathaniel Grammel is a PhD student of computer science at the University of Maryland, College Park. His research interests include online algorithms and online matching.

Will Ma is an associate professor at the Graduate School of Business and Data Science Institute of Columbia University. He works on the theory of online algorithms and their applications in revenue and supply chain management.

Aravind Srinivasan is a Distinguished University Professor and professor of computer science at the University of Maryland, College Park. His interests include algorithms, probabilistic methods, optimization, data science, and healthcare applications.