Discrete Optimization 48 (2023) 100778

Contents lists available at ScienceDirect

Discrete Optimization

www.elsevier.com/locate/disopt

Secretary and online matching problems with machine learned advice

Antonios Antoniadis [a,1], Themis Gouleakis [b], Pieter Kleer [b,*,2], Pavel Kolev [b]

[a] *Saarland University and Max Planck Institute for Informatics, Saarbrücken, Germany*
[b] *Max Planck Institute for Informatics, Saarbrücken, Germany*

ARTICLE INFO

Article history:
Received 19 May 2021
Received in revised form 10 December 2022
Accepted 8 May 2023
Available online 9 June 2023

Keywords:
Secretary problem
Online bipartite matching
Machine learned advice
Learning augmentation

ABSTRACT

The classic analysis of online algorithms, due to its worst-case nature, can be quite pessimistic when the input instance at hand is far from worst-case. In contrast, machine learning approaches shine in exploiting patterns in past inputs in order to predict the future. However, such predictions, although usually accurate, can be arbitrarily poor. Inspired by a recent line of work, we augment three well-known online settings with machine learned predictions about the future, and develop algorithms that take these predictions into account. In particular, we study the following online selection problems: (i) the classic secretary problem, (ii) online bipartite matching and (iii) the graphic matroid secretary problem. Our algorithms still come with a worst-case performance guarantee in the case that predictions are subpar while obtaining an improved competitive ratio (over the best-known classic online algorithm for each problem) when the predictions are sufficiently accurate. For each algorithm, we establish a trade-off between the competitive ratios obtained in the two respective cases.

1. Introduction

In this work, we consider various online selection algorithms augmented with so-called *machine learned advice*. In particular, we consider secretary and online bipartite matching problems. The high-level idea is to incorporate some form of *predictions* in an existing online algorithm in order to get the best of two worlds: (i) provably improve the algorithm's performance guarantee in the case that predictions are sufficiently good, while (ii) losing only a constant factor of the algorithm's existing worst-case performance guarantee, when the predictions are subpar. Improving the performance of classic online algorithms with the help of machine learned predictions, e.g., in the sense of (i) and (ii), is a relatively new area that has gained a lot of attention

* Corresponding author.
E-mail addresses: a.antoniadis@utwente.nl (A. Antoniadis), a.antoniadis@utwente.nl (T. Gouleakis), p.s.kleer@tilburguniversity.edu (P. Kleer), p.s.kleer@tilburguniversity.edu (P. Kolev).
[1] University of Twente, The Netherlands.
[2] Tilburg University, The Netherlands.

https://doi.org/10.1016/j.disopt.2023.100778

in the last couple of years [1–9] motivated by the tremendous surge in the field of machine learning. See also the recent overview article of Mitzenmacher and Vassilvitskii [10] or the website [11] that collects work on algorithms with predictions.

We motivate the idea of incorporating machine-learned advice, in the class of problems studied in this work, by means of a simple real-world problem. Consider the following setting for selling a laptop on an online platform.[3] A known number of potential buyers arrive one by one, say, in a uniformly random order, and report a price that they are willing to pay for the laptop. Whenever a buyer arrives, we have to irrevocably decide if we want to sell at the given price, or wait for a better offer. Based on historical data, e.g., regarding previous online sales of laptops with similar specs, the online platform might suggest a (machine learned) prediction for the maximum price that some buyer is likely to offer for the laptop.

How can we exploit this information in our decision process? One problem that arises here is that we do not have any formal guarantees for how accurate the machine-learned advice is for any particular instance. For example, suppose we get a prediction of 900 dollars as the maximum price that some buyer will likely offer. One extreme policy is to blindly trust this prediction and wait for the first buyer to come along that offers a price sufficiently close to 900 dollars. If this prediction is indeed accurate, this policy has an almost perfect performance guarantee, in the sense that we will sell to the (almost) highest bidder. However, if the best offer is only, say, 500 dollars, we will never sell to this buyer (unless this offer arrives last), since the advice is to wait for a better offer to come along. In particular, the performance guarantee of this selling policy depends on the *prediction error* (400 dollars in this case) which can become arbitrarily large. The other extreme policy is to completely ignore the prediction of 900 dollars and just run the classic secretary algorithm: Observe a $1/e$-fraction of the buyers and remember the highest price seen in this fraction. After that, sell to the first buyer who is willing to pay more than the best seen price in the first $1/e$-fraction. This yields, in expectation, a selling price of at least $1/e$ times the highest offer [13,14].

Can we somehow combine the preceding two extreme selling-policies, so that we get a performance guarantee strictly better than that of $1/e$ in the case where the prediction for the highest offer is not too far off, while not losing too much over the guarantee of $1/e$ otherwise? Note that (even partially) trusting poor predictions often comes at a price, and thus obtaining a competitive ratio worse than $1/e$ seems inevitable in this case. We show that there is in fact a trade-off between the competitive ratio that we can achieve when the prediction is accurate and the one we obtain when the prediction error turns out to be large.

1.1. Our models and contributions

We show how one can incorporate predictions in various online selection algorithms for problems that generalize the classic secretary problem. The overall goal is to include *as little predictive information as possible* into the algorithm, while still obtaining improvements in the case that the information is accurate. Our results are parameterized by (among other parameters) the so-called prediction error η that measures the quality of the given predictions.

We briefly sketch each of the problems studied in this work, and then conclude with the description of Meta result 1.1, that applies to all of them.

Secretary problem. In order to illustrate our ideas and techniques, we start by augmenting the classic secretary problem with predictions. We emphasize that we consider the so-called *value maximization* version of the problem. For details, see Section 3. Here, we are given a prediction p^* for the maximum value among all arriving secretaries. This corresponds to a prediction for the maximum price somebody is willing to offer in the laptop example. The prediction error is then defined as $\eta = |p^* - v^*|$, where v^* is the true maximum value among all secretaries. We emphasize that the algorithm is not aware of the prediction error η, and this parameter is only used to analyze the algorithm's performance guarantee.

[3] This example is similar to the classic secretary problem [12].

Online bipartite matching with vertex arrivals. In Section 4, we study the online bipartite matching problem on a bipartite graph $G = (L \cup R, E)$, with $|L| = n$ and $|R| = m$. The vertex set R is known from the start while the nodes in the set L arrive *online* in a uniformly random order [15,16]. Upon arrival, a node reveals the edge weights to its neighbors in R. We have to irrevocably decide if we want to match up the arrived online node with one of its (currently unmatched) neighbors in R. Kesselheim et al. [16] gave a tight $1/e$-competitive deterministic algorithm for this setting that significantly generalizes the same guarantee for the classic secretary algorithm [13,14]. Note that the classic secretary problem corresponds to the case in which there is one offline node, i.e., $|R| = 1$.

The prediction that we consider in this setting is a vector of values $p^* = (p_1^*, \ldots, p_m^*)$ that predicts the edge weights adjacent to the nodes $r \in R$ in some fixed optimal (offline) bipartite matching in G. That is, the prediction p^* indicates the existence of a fixed optimal bipartite matching in which each node $r \in R$ is adjacent to an edge with weight p_r^*. The prediction error is then the maximum prediction error taken over all nodes in $r \in R$ and minimized over all optimal matchings. This generalizes the prediction used for the classic secretary problem.

An interpretation of this problem can be found in the problem where we want to sell a number of items (the offline nodes) to customers that arrive online one-by-one. Upon arrival a customer can be assigned at most one item. The prediction can be interpreted as an estimate of the value for which an item is typically sold in an optimal offline solution. From a theoretical point of view, our prediction setting is closely related to the *vertex-weighted online bipartite matching problem* [17], which will be discussed in Section 4.

Graphic matroid secretary problem. In Section 5, we augment the graphic matroid secretary problem with predictions. In this problem, the edges of a given undirected graph $G = (V, E)$, with $|V| = n$ and $|E| = m$, arrive in a uniformly random order. The goal is to select a subset of edges of maximum weight under the constraint that this subset is a forest. That is, it is not allowed to select a subset of edges that form a cycle in G. This problem also generalizes the classic secretary problem by considering a graph with k parallel edges between two nodes. The best known algorithm for this online problem is a (randomized) 1/4-competitive algorithm by Soto, Turkieltaub and Verdugo [18]. Their algorithm proceeds by first selecting no elements from a prefix of the sequence of elements with randomly chosen size, followed by selecting an element if and only if it belongs to a "canonically computed" (see [18] for details) offline optimal solution, and can be added to the set of elements currently selected online.

The prediction that we consider here is a vector of values $p = (p_1^*, \ldots, p_n^*)$ where p_i^* predicts the maximum edge weight that node $i \in V$ is adjacent to, in the graph G. This is equivalent to saying that p_i^* is the maximum edge weight adjacent to node $i \in V$ in a given optimal spanning tree (we assume that G is connected for sake of simplicity), which is, in a sense, in line with the predictions used in Section 4. (We note that the predictions model the optimal spanning tree in the case when all edge-weights are pairwise distinct. Otherwise, there can be many (offline) optimal spanning trees, and thus the predictions do not encode a unique optimal spanning tree. We intentionally chose not to use predictions regarding which edges are part of an optimal solution, as in our opinion, such an assumption would be too strong.) The prediction error is defined as the maximum prediction error over the individual nodes.

As a result of possible independent interest, we show that there exists a *deterministic* $(1/4 - o(1))$-competitive algorithm for the graphic matroid secretary problem, which can roughly be seen as a deterministic version of the algorithm of Soto et al. [18]. Alternatively, our algorithm can be seen as a variation on the (deterministic) algorithm of Kesselheim et al. [16] for the case of online bipartite matching, in combination with an idea introduced in [19].

Meta result

We note that for all the problems above, one cannot hope for an algorithm with a performance guarantee better than $1/e$ in the corresponding settings without predictions, as this bound is known to be optimal already for the classic secretary problem [13,14] (recall that the other two problems we consider are generalizations of the classic secretary problem). Hence, our goal is to design algorithms that improve upon the $1/e$ worst-case competitive ratio in the case where the prediction error is sufficiently small, and otherwise (when the prediction error is large) never lose more than a constant (multiplicative) factor over the worst-case competitive ratio.

For each of the preceding three problems, we augment existing algorithms with predictions. All of our resulting algorithms are *deterministic*. We show that the canonical approaches for the secretary problem [13,14] and the online bipartite matching problem [16] can be naturally augmented with predictions. We also demonstrate how to adapt our novel deterministic algorithm for the graphic matroid secretary problem. Further, we comment on randomized approaches for the three problems in each respective section.

Meta result 1.1. *There is a polynomial time deterministic algorithm that incorporates the predictions p^* such that for some constants $0 < \alpha, \beta < 1$ it is*

(i) α-competitive with $\alpha > \frac{1}{e}$, when the prediction error is sufficiently small; and
(ii) β-competitive with $\beta < \frac{1}{e}$, independently of the prediction error.

We note that there is a correlation between the constants α and β, which can be intuitively described as follows: The more one is willing to give up in the worst-case guarantee, i.e. the more confidence we have in the predictions, the better the competitive ratio becomes in the case where the predictions are sufficiently accurate.

We next give a high-level overview of our approach in order to establish Meta result 1.1 for our three problems. In all cases we split up the random arrival sequence in three phases. In the first phase, we merely observe the arriving elements without selecting any of them; this is called the observation (or sampling) phase. In the remaining two phases, we run two extreme policies: One that fully exploits the predictions, and one that ignores them completely. Although both extreme policies can be analyzed individually using existing techniques, it is a non-trivial task to show that, when combined deterministically, they do not obstruct each other too much. For example, suppose that the second phase tries to exploit the predictions, but that the predictions are very poor. We might select many "bad" elements in the second phase which makes it impossible to select sufficiently many good elements in the third phase in order to still obtain a β-competitive algorithm. In some cases, the execution order of the two extreme policies is crucial for the analysis of the approximation guarantee.

Some of the issues sketched in the previous paragraph can be circumvented by using *randomized* algorithms. For example, one can instead randomize over the two policies that either exploit or ignore the predictions, respectively. This avoids the problem of making sure that the two policies "do not obstruct each other too much". In this work, we only focus on *deterministic algorithms* as the best online algorithms for the secretary, online bipartite matching and graphic matroid secretary problem are of a deterministic nature (for the third problem we show that in this work). We will elaborate more on this in the respective sections to come. In a nutshell, for the secretary problem, our deterministic algorithm has a better performance guarantee than the "straightforward" randomization between the two extreme policies. For the online bipartite matching and graphic matroid secretary problem, a randomization of this type performs (in expectation) better than our deterministic algorithms. It is in an interesting open problem to find deterministic algorithms with better performance guarantees.

We give detailed formulations of Meta result 1.1: in Theorem 3.1 for the secretary problem; in Theorem 4.1 for the online bipartite matching problem; and in Theorem 5.3 for the graphic matroid secretary problem.

Remark 1.2. In the statements of Theorems 3.1, 4.1 and 5.3 it is assumed that the set of objects O arriving online (either vertices or edges) is asymptotically large. We hide $o(1)$-terms, with respect to $n = |O|$, at certain places for the sake of readability.

Although the predictions provide relatively little information about the optimal solution, we are still able to obtain improved theoretical guarantees in the case where the predictions are sufficiently accurate. In the online bipartite matching setting with predictions for the nodes in R, we can essentially get close to a 1/2-approximation assuming the predictions are close to perfect. This matches the state-of-the-art of what one can hope for with perfect predictions, because of the relation with the so-called *vertex-weighted online bipartite matching problem*, which will be explained in Section 4. The best known deterministic algorithm for this problem, to the best of our knowledge, is a simple 1/2-competitive greedy algorithm [17]. Roughly speaking, our algorithm converges to that in [17] when the predictions get close to perfect. This will be discussed further in Section 4. For the graphic matroid secretary problem, we are also able to get close to a 1/2-approximation in the case where the predictions (the maximum edge weights adjacent to the nodes in the graph) get close to perfect. We note that this is probably not tight. We suspect that, when given perfect predictions, it is possible to obtain an algorithm with a better approximation guarantee. This is an interesting open problem.

1.2. Related work

This subsection consists of three parts. First we give a short overview of related problems that have been analyzed with the inclusion of machine learned advice following the frameworks in [5,6]. We continue with relevant approximation algorithms for the matroid secretary problem, and then we consider models that incorporate additional information, such as prior distributions.

Black-box machine learned advice. Although online algorithms with machine learned advice are a relatively new area, there has already been a number of interesting results. We note that most of the following results are analyzed by means of *consistency* (competitive-ratio in the case of perfect predictions) and *robustness* (worst-case competitive-ratio regardless of prediction quality), but the precise formal definitions of consistency and robustness slightly differ in each paper [5,6].[4] Our results can also be interpreted within this framework, but for the sake of completeness we give the competitive ratios as a function of the prediction error. Purohit et al. [6], considered the *ski rental problem* and the *non-clairvoyant scheduling problem*. For both problems they gave algorithms that are both consistent and robust, and with a flavor similar to ours, the trade-off between the robustness and consistency of their algorithms are given as a function of some hyperparameter which has to be chosen by the algorithm in advance. For the ski rental problem, in particular, Gollapudi et al. [7] considered the setting with multiple predictors, and they provided tight algorithms.

Lykouris and Vassilvitskii [5] studied the *caching problem* (also known in the literature as *paging*), and were able to adapt the classic Marker algorithm [21] to obtain a trade-off between robustness and consistency, for this problem. Rohatgi [3] subsequently gave an algorithm whose competitive ratio has an improved dependence on the prediction errors.

Further results in online algorithms with machine learned advice include the work by Lattanzi et al. [4] who studied the *restricted assignment scheduling problem*, and the work by Mitzenmacher [8] who considered a different *scheduling/queuing problem*. They introduced a novel quality measure for evaluating algorithms, called the *price of misprediction*.

Mahdian et al. [22] studied problems where it is assumed that there exists an optimistic algorithm (which could in some way be interpreted as a prediction), and designed a meta-algorithm that interpolates between a worst-case algorithm and the optimistic one. They considered several problems, including the *allocation of online advertisement space*, and for each gave an algorithm whose competitive ratio is also an interpolation between the competitive ratios of its corresponding optimistic and worst-case algorithms. However, the performance guarantee is not given as a function of the prediction error, but rather only as a function of the respective ratios and the interpolation parameter.

[4] The term "robustness" has also been used in connection with secretary problems (see for example [20]), but in a totally different sense.

Approximation algorithms for the matroid secretary problem. The classic secretary problem was originally introduced by Gardner [12], and solved by Lindley [13] and Dynkin [14], who gave $1/e$-competitive algorithms. Babaioff et al. [19] introduced the matroid secretary problem, a considerable generalization of the classic secretary problem, where the goal is to select a set of secretaries with maximum total value under a matroid constraint for the set of feasible secretaries. They provided an $O(1/\log(r))$-competitive algorithm for this problem, where r is the rank of the underlying matroid. Lachish [23] later gave an $O(1/\log\log(r))$-competitive algorithm, and a simplified algorithm with the same guarantee was given by Feldman, Svensson and Zenklusen [24]. It is still a major open problem if there exists a constant-competitive algorithm for the matroid secretary problem. Nevertheless, many constant-competitive algorithms are known for special classes of matroids, and we mention those relevant to the results in this work (see, e.g., [18,19] for further related work).

Babaioff et al. [19] provided a 1/16-competitive algorithm for the case of transversal matroids,[5] which was later improved to a 1/8-competitive algorithm by Dimitrov and Plaxton [25]. Korula and Pál [15] provided the first constant competitive algorithm for the online bipartite matching problem considered in Section 4, of which the transversal matroid secretary problem is a special case. In particular, they gave a 1/8-approximation. Kesselheim et al. [16] provided a $1/e$-competitive algorithm, which is best possible, as discussed above.

For the graphic matroid secretary problem, Babaioff et al. [19] provide a deterministic 1/16-competitive algorithm. This was improved to a $1/(3e)$-competitive algorithm by Babaioff et al. [26]; a $1/(2e)$-competitive algorithm by Korula and Pál [15]; and a 1/4-competitive algorithm by Soto et al. [18], which is currently the best algorithm. The algorithm from [19] is deterministic, whereas the other three are randomized. All algorithms run in polynomial time.

Other models, extensions and variations. There is a vast literature on online selection algorithms for problems similar to the (matroid) secretary problem. Here we discuss some recent directions and other models incorporating some form of prior information. These problems are concerned with relaxing assumptions of existing problems, rather than "augmenting" them with some form of advice.

The most important assumption in the secretary model that we consider is the fact that elements arrive in a uniformly random order. If the elements arrive in an adversarial order, there is not much one can achieve: There is a trivial randomized algorithm that selects every element with probability $1/n$, yielding a $1/n$-competitive algorithm; deterministically no finite competitive algorithm is possible with a guarantee independent of the values of the elements. There has been a recent interest in studying *intermediate* arrival models that are not completely adversarial, nor uniformly random. Kesselheim, Kleinberg and Niazadeh [27] study non-uniform random arrival orderings under which (asymptotically) one can still obtain a $1/e$-competitive algorithm for the secretary problem. Bradac et al. [20] consider the so-called Byzantine secretary model in which some elements arrive uniformly at random, but where an adversary controls a set of elements that can be inserted in the ordering of the uniform elements in an adversarial manner. See also the very recent work of Garg et al. [28] for a conceptually similar model.

In a slightly different setting, Kaplan et al. [29] consider a secretary problem with the assumption that the algorithm has access to a random sample of the adversarial distribution ahead of time. For this setting they provide an algorithm with almost tight competitive-ratio for small sample-sizes. Related models are studied in [30,31].

Furthermore, there is also a vast literature on so-called *prophet inequalities*. In the basic model, the elements arrive in an adversarial order, but there is a prior distributional information given for the values of the elements $\{1, \dots, n\}$. That is, one is given probability distributions $X_1, \dots, X_n$ from which the values

[5] The transversal matroid secretary problem is a special case of the online bipartite matching problem where every node $i \in L$ that arrives online has one common edge weight w_i on all edges $\{i, j\} \in E$.

of the elements are drawn. Upon arrival of an element e, its value drawn according to X_e is revealed and an irrevocable decision is made whether to select this element or not. Note that the available distributional information can be used to decide on whether to select an element. The goal is to maximize the expected value, taken over all prior distributions, of the selected element. For surveys on recent developments, refer to [32,33]. Here we discuss some classic results and recent related works. Krengel, Sucheston and Garling [34] show that there is an optimal 1/2-competitive algorithm for this problem. Kleinberg and Weinberg [35] gave a significant generalization of this result to matroid prophet inequalities, where multiple elements can be selected subject to a matroid feasibility constraint (an analogue of the matroid secretary problem). There is also a growing interest in the prophet secretary problem [36], in which the elements arrive uniformly random (as in the secretary problem); see also [33].

Recently, settings with more limited prior information gained a lot of interest. These works address the quite strong assumption of knowing all element-wise prior distributions. Azar, Kleinberg and Weinberg [37] study the setting in which one has only access to one sample from every distribution, as opposed to the whole distribution; see also [38]. Correa et al. [39] study this problem under the assumption that all elements are identically distributed. Recently, an extension of this setting was considered by Correa et al. [40]. Furthermore, Dütting and Kesselheim [41] consider prophet inequalities with inaccurate (measured in terms of a metric) prior distributions $\tilde{X}_1, \dots, \tilde{X}_n$, while the true distributions $X_1, \dots, X_n$ remain unknown. They study to what extent the existing algorithms are robust against inaccurate prior distributions.

Although our setting also assumes additional information about the input instance, there are major differences. Mainly, we are interested in including a minimal amount of predictive information about an optimal (offline) solution, which yields a quantitative improvement in the case where the prediction is sufficiently accurate. This is a completely different assumption than having a priori element-wise (possibly inaccurate) probability distributions. Furthermore, our setting does not assume that the predictive information necessarily comes from a distribution (which is then used to measure the expected performance of an algorithm), but can be obtained in a more general fashion from historical data (using, e.g., statistical or machine learning techniques). Finally, and in contrast to other settings, the information received in our setting can be inaccurate (and this is non-trivial do deal with).

Also conceptually close to our setting is the work of Dütting et al. [42]. They consider a general framework for incorporating various forms of "advice" in the classic secretary model capturing e.g. sample-based or signal information. They derive various optimal policies in their framework, however, it does not cover the machine-learned advice model built on the concepts of *consistency* and *robustness* [5,6] discussed earlier.

2. Preliminaries

In this section we formally define the online algorithms of interest, provide the necessary graph theoretical notation, and define the so-called Lambert W-function that will be used in Section 3.

2.1. Online algorithms with uniformly random arrivals

We briefly sketch some relevant definitions for the online problems that we consider in this work. We consider online selection problems in which the goal is to select the "best feasible" subset out of a finite set of objects O with size $|O| = n$, that arrive online in a uniformly random order. More formally, the n objects are revealed to the algorithm one object per round. In each round i, and upon revelation of the current object $o_i \in O$, the online selection algorithm has to irrevocably select an *outcome* z_i out of a set of possible outcomes $Z(o_i)$ (which may depend on $o_1, o_2, \dots o_i$ as well as $z_1, z_2, \dots z_{i-1}$.) Each outcome z_i is associated with a *value* $v_i(z_i)$, and all values v_i become known to the algorithm with the arrival of o_i. The goal is to maximize the total value $T = \sum_i v_i(z_i)$.

The value of an algorithm $\mathcal{A}$ selecting outcomes $z_1, z_2 \dots z_n$ on input sequence $\sigma = (o_1, \dots, o_n)$ is defined as $T(\mathcal{A}(\sigma)) = \sum_i v_i(z_i)$. We sometimes refer to the input sequence as a *permutation* or *arrival order*. Such an algorithm $\mathcal{A}$ is γ-competitive if $\mathbb{E}(T(\mathcal{A}(\sigma))) \geq \gamma \cdot \mathrm{OPT}(\sigma)$, for $0 < \gamma \leq 1$, where $\mathrm{OPT}(\sigma)$ is the objective value of an offline optimal solution, i.e., the solution of an algorithm that is aware of the whole input sequence σ in advance. The expectation is taken over the randomness in σ (and the internal randomness of $\mathcal{A}$ in case of a randomized algorithm). Alternatively, we say that $\mathcal{A}$ is a γ-approximation.

2.2. Graph theoretical notation

An undirected graph $G = (V, E)$ is defined by a set of nodes V and set of edges $E \subseteq \{\{u, v\} : u, v \in V, u \neq v\}$. A bipartite graph $G = (L \cup R, E)$ is given by two sets of nodes L and R, and $E \subseteq \{\{\ell, r\} : \ell \in L, r \in R\}$. In the bipartite case we sometimes write (ℓ, r) instead of $\{\ell, r\}$ in order to indicate that $\ell \in L$ and $r \in R$, i.e., we give a direction to the edge. We also use this notation for directed arcs in general directed graphs. For a set of nodes W, we use $G[W]$ to denote the induced (bipartite) subgraph on the nodes in W.

A function $w : E \to \mathbb{R}_{\geq 0}$ is called a weight function on the edges in E; we sometimes write $w(u, v)$ or w_{uv} in order to denote $w(\{u, v\})$ for $\{u, v\} \in E$. For a collection $F \subseteq E$ of edges, we write $w(F) = \sum_{\{u,v\} \in F} w(u, v)$. A matching $M \subseteq E$ is a subset of edges so that every node is adjacent to at most one edge in M. For a set of nodes W, we write $W[M]$ to denote the nodes in W that are adjacent to an edge in M. Such nodes are said to be matched. If G is undirected, we say that M is perfect if every node in V is adjacent to precisely one edge in M. If G is bipartite, we say that M is perfect with respect to L if every $\ell \in L$ is adjacent to one edge in M, and perfect with respect to R if every $r \in R$ is adjacent to some edge in M.

Remark 2.1. When G is bipartite, we will assume that for every subset $S \subseteq L$, there is a perfect matching with respect to S in $G[S \cup R]$. For our applications, this can be done without loss of generality by adding for every $\ell \in L$ a node r^ℓ to the set R, and adding the edge $\{\ell, r^\ell\}$ to E. Moreover, given a weight function on E we extend it to a weight function on E' by giving all the new edges weight zero.

2.3. Lambert W-function

The Lambert W-function is the inverse relation of the function $f(w) = we^w$. Here, we consider this function over the real numbers, i.e., the case $f : \mathbb{R} \to \mathbb{R}$. Consider the equation $ye^y = x$. For $-1/e \leq x < 0$, this equation has two solutions denoted by $y = W_{-1}(x)$ and $y = W_0(x)$, where $W_{-1}(x) \leq W_0(x)$ with equality if and only if $x = -1/e$.

3. Secretary problem

In the secretary problem there is a set $\{1, \dots, n\}$ of secretaries, each with a value $v_i \geq 0$ for $i \in \{1, \dots, n\}$, that arrive in a *uniformly random order*. Whenever a secretary arrives, we have to irrevocably decide whether we want to hire that person. If we decide to hire a secretary, we automatically reject all subsequent candidates. The goal is to select the secretary with the highest value. We assume without loss of generality that all values v_i are distinct. This can be done by introducing a suitable tie-breaking rule if necessary.

There are two versions of the secretary problem. In the *classic secretary problem*, the goal is to maximize the probability with which the best secretary is chosen. We consider a different version, where the goal is to maximize the expected value of the chosen secretary. We refer to this as the *value-maximization* *secretary problem.*[6] In the remainder of this work, the term 'secretary problem' will always refer to the value-maximization secretary problem, unless stated otherwise.

The machine learned advice that we consider in this section is a prediction p^* for the maximum value $\mathrm{OPT} = \max_i v_i$ among all secretaries.[7] Note that we do not predict which secretary has the highest value. We define the *prediction error* as

$$\eta = |p^* - \mathrm{OPT}|.$$

We emphasize that this parameter is not known to the algorithm, but is only used to analyze the algorithm's performance guarantee.

3.1. Deterministic algorithm

We will next describe our algorithm that incorporates the prediction p^* for the maximum value. Henceforth, we will refer to secretaries as elements. The formal description is given in Algorithm 1. We introduce two (hyper)parameters c and λ that can, roughly speaking, be used to control the robustness-consistency trade-off of Algorithm 1 in the spirit of [5,6]. Algorithm 1 is a generalization of the well-known optimal algorithm for both the classic and value-maximization secretary problem [13,14], which can be obtained by setting $c = 1$, in which case there is no Phase II. The (optimal) solution [13,14] to the secretary problem without predictions is to first observe a fraction of n/e elements which we call Phase I. After that, in Phase II, the first element with a value higher than the best value seen in Phase I is selected.

When looking closer at the analysis in [13,14] it becomes clear that one can obtain a $1/(ce)$-competitive algorithm, for $c \geq 1$ by sampling either less or more elements in Phase I, more specifically a $w_{-1} = \exp\{W_{-1}(-1/(ce))\}$ or $w_0 = \exp\{W_0(-1/(ce))\}$ fraction of the elements respectively. Here W_{-1} and W_0 are the branches of the Lambert W-function (see Section 2.3). Based on w_{-1} and w_0, we define a new algorithm consisting of three phases: Phase I is again an observation phase in which we do not select any element; in Phase II we will try to exploit the prediction; and in Phase III we ignore the prediction, essentially running the classic secretary algorithm with a different observation phase, in order to still guarantee a $1/(ce)$-competitive algorithm in case the prediction is very bad. Roughly speaking, the parameter c models the factor that we are willing to lose in the worst-case when the prediction turns out to be poor.

How to exploit the prediction in Phase II? If we would *a priori* know that the prediction error is small, then it seems reasonable to pick the first element that has a value 'close enough' to the predicted optimal value. An issue that arises here is that we do not know whether the predicted value is smaller or larger than the actual optimal value. In the latter case, the entire Phase II of the algorithm would be rendered useless, even if the prediction error was arbitrarily small. In order to circumvent this issue, one can first lower the predicted value p^* slightly by some $\lambda > 0$ and then select the first element that is greater or equal than the threshold $p^* - \lambda$. Roughly speaking, the parameter λ can be interpreted as a guess for the prediction error η. (Alternatively, one could define an interval around p^*, but given that we get a prediction for the maximum value, this does not make a big difference.)

The input parameter λ can be seen as a confidence parameter in the prediction p^* that allows us to interpolate between the following two extreme cases:

(i) If we have very low confidence in the prediction, we choose λ close to p^*;
(ii) If we have high confidence in the prediction, we choose λ close to 0.

[6] Any α-approximation for the classic secretary problem yields an α-approximation for the value-maximization variant.

[7] Such a prediction does not seem to have any relevance in case the goal is to maximize the probability with which the best secretary is chosen. Intuitively, for every instance $(v_1, \ldots, v_n)$ there is a 'more or less isomorphic' instance $(v_1', \ldots, v_n')$ for which $v_i' < v_j'$ if and only if $v_i < v_j$ for all i, j, and for which, all values are close to each other and also to the prediction p^*, for which we assume that $p^* < \min_i v_i'$. Any algorithm that uses only pairwise comparisons between the values v_i' can, intuitively, not benefit from the prediction p^*. Of course, for the value-maximization variant, choosing any i will be close to optimal in this case if the prediction error is small.

ALGORITHM 1: Value-maximization secretary algorithm

Input : Prediction p^* for (unknown) value $\max_i v_i$; confidence parameter $0 \leq \lambda \leq p^*$ and $c \geq 1$.
Output: Element a.

```
Set v' = 0.
Phase I:
for i = 1, ..., ⌊exp{W_{-1}(-1/(ce))} · n⌋ do
  | Set v' = max{v', v_i}
end
Set t = max{v', p* - λ}.
Phase II:
for i = ⌊exp{W_{-1}(-1/(ce))} · n⌋ + 1, ..., ⌊exp{W_0(-1/(ce))} · n⌋ do
  if v_i > t then
    | Select element a_i and STOP.
  end
end
Set t = max{v_j : j ∈ {1, ..., ⌊exp(W_0(-1/(ce))) · n⌋}}.
Phase III:
for i = ⌊exp{W_0(-1/(ce))} · n⌋ + 1, ..., n do
  if v_i > t then
    | Select element a_i and STOP.
  end
end
```

In the first case, we essentially get back the classic solution [13,14] but now with a guarantee of $1/(ce)$. Otherwise, when the confidence in the prediction is high, we get a competitive ratio better than $1/e$ in case the prediction error η is in fact small, in particular, smaller than λ. If our confidence in the prediction turned out to be wrong, when λ is larger than the prediction error, we still obtain a $1/(ce)$-competitive algorithm. Recall that the parameter c models what factor we are willing to lose in the worst-case when the prediction is poor (but the confidence in the prediction is high). In Theorem 3.1 below, we analyze Algorithm 1.

Theorem 3.1. *For any $\lambda \geq 0$ and $c > 1$, there is a deterministic algorithm for the (value-maximization) secretary problem that is asymptotically $g_{c,\lambda}(\eta)$-competitive in expectation, where*

$$g_{c,\lambda}(\eta) = \begin{cases} \max\left\{\frac{1}{ce}, \left[f(c)\left(\max\left\{1 - \frac{\lambda+\eta}{OPT}, 0\right\}\right)\right]\right\} & \text{if } 0 \leq \eta < \lambda \\ \frac{1}{ce} & \text{if } \eta \geq \lambda \end{cases},$$

and the function $f(c)$ is given in terms of the two branches W_0 and W_{-1} of the Lambert W-function and reads

$$f(c) = \exp\{W_0(-1/(ce))\} - \exp\{W_{-1}(-1/(ce))\}.$$

We note that λ and c are independent parameters that provide the most general description of the competitive ratio. Recall that λ is our confidence in the prediction and c describes how much we are willing to lose in the worst case. Although these parameters can be set independently, some combinations of them are not very sensible, as one might not get an improved performance guarantee, even when the prediction error is small (for instance, if $c = 1$, i.e., we are not willing to lose anything in the worst case, then it is not helpful to consider the prediction at all). To illustrate the influence of these parameters on the competitive ratio, in Fig. 1, we plot various combinations of the input parameters c, λ and p^* of Algorithm 1, assuming that $\eta = 0$. In this case $p^* = \text{OPT}$ and the competitive ratio simplifies to

$$g_{c,\lambda}(0) = \max\left\{\frac{1}{ce},\ f(c) \cdot \max\left\{1 - \frac{\lambda}{p^*}, 0\right\}\right\}.$$

We therefore choose the axes of Fig. 1 to be λ/p^* and c.

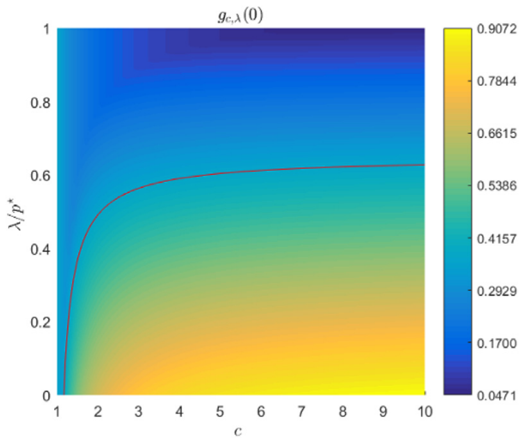

Fig. 1. The red curve shows the optimal competitive ratio without predictions, i.e., $g_{c,\lambda}(0) = 1/e$. Our algorithm achieves an improved competitive ratio $g_{c,\lambda}(0) > 1/e$ in the area below this curve, and a worse competitive ratio $g_{c,\lambda}(0) < 1/e$ in the area above it.

Furthermore, as one does not know the prediction error η, there is no way in choosing these parameters optimally, since different prediction errors require different settings of λ and c.

To get an impression of the statement in Theorem 3.1, if we have, for example, $\eta + \lambda = \frac{1}{10}\mathrm{OPT}$, then we start improving over $1/e$ for $c \geq 1.185$. Moreover, if one believes that the prediction error is low, one should set c very high (hence approaching a 1-competitive algorithm in case the predictions are close to perfect).

Remark 3.2. The fact that OPT appears in the competitive ratio in Theorem 3.1 stems from the definition of the competitive ratio in combination with the fact that η is defined as the difference between the optimal and the predicted values. In particular, a specific η could refer to a tiny or a huge relative error depending on the value of OPT, so it is not surprising that the competitive ratio is expressed in terms of η/OPT.

Remark 3.3. Note, that the bound obtained in Theorem 3.1 has a discontinuity at $\eta = \lambda$. This can be easily smoothed out by selecting λ according to some distribution, which now represents our confidence in the prediction p^*. The competitive ratio will start to drop earlier in this case, and will continuously reach $1/(ce)$. Furthermore, for $\eta = \lambda = 0$ this bound is tight for any fixed c. A further illustration of how the competitive ratio changes as a function of η is given in Appendix A.

Proof of Theorem 3.1. By carefully looking into the analysis of the classic secretary problem, see, e.g., [13,14], it becomes clear that although sampling an $1/e$-fraction of the items is the optimal trade-off for the classic algorithm and results in a competitive ratio of $1/e$, one could obtain a $1/(ce)$-competitive ratio (for $c > 1$) in two ways: by sampling either less, or more items, more specifically an $\exp\{W_{-1}(-1/(ce))\}$ or $\exp\{W_0(-1/(ce))\}$ fraction of the items respectively. These quantities arise as the two solutions of the equation

$$-x \ln x = \frac{1}{ce}.$$

We next provide two lower bounds on the competitive ratio.

First of all, we prove that in the worst-case we are always $1/(ce)$-competitive. We consider two cases.

Case 1: $p^* - \lambda > OPT$. Then we never pick an element in Phase II, which means that the algorithm is equivalent to the algorithm that observes a fraction $\exp\{W_0(-1/(ce))\}$ of all elements and then chooses the first element better than what has been seen before, which we know is $1/(ce)$-competitive.

Case 2. $p^* - \lambda \leq OPT$. Let "Algorithm A" be the algorithm that first observes a fraction $\exp\{W_{-1}(-1/(ce))\}$ of all elements and then selects the first element better than what has been seen before (which we know is $1/(ce)$-competitive [13,14]). Consider a fixed arrival order and suppose that, for this permutation, we select an element with value OPT in Algorithm A. Our algorithm also chooses an element of value OPT in this case, because of the condition $p^* - \lambda \leq OPT$. As the analysis in [13,14] for the classic secretary problem relies on analyzing the probability with which we pick an element of value OPT, it follows that our algorithm is also $1/(ce)$-competitive in this case.

The second bound on the competitive ratio applies to cases in which the prediction error is small. In particular, suppose that $0 \leq \eta < \lambda$.

Case 1: $p^* > OPT$. We know that $p^* - \lambda < OPT$, as $\eta < \lambda$. Therefore, if OPT appears in Phase II, and we have not picked anything so far, we will pick OPT. Since OPT appears in Phase II with probability $f(c)$, we in particular pick some element in Phase II with value at least $OPT - \lambda$ with probability $f(c)$ (note that this element does not have to be OPT necessarily).

Case 2: $p^* \leq OPT$. In this case, using similar reasoning as in Case 1, with probability $f(c)$ we will pick some element with value at least $OPT - \lambda - \eta$. To see this, note that in the worst case we would have $p^* = OPT - \eta$, and we could select an element with value $p^* - \lambda$, which means that the value of the selected item is $OPT - \lambda - \eta$.

This means that, in any case, with probability at least $f(c)$, we will pick some element in Phase II with value at least $\min\{OPT - \lambda, OPT - \lambda - \eta\} = OPT - \lambda - \eta$ if $\eta < \lambda$. That is, if $0 \leq \eta < \lambda$, and if we assume that $\mathrm{OPT} - \lambda - \eta \geq 0$, we are guaranteed to be $f(c)\left(1 - (\lambda + \eta)/\mathrm{OPT}\right)$-competitive. □

3.2. Straightforward randomization

One natural question that arises from the bound in Theorem 3.1 is whether one can significantly improve the result using randomization (we also already discussed randomization as a way of "smoothing" out the discontinuity at $\eta = \lambda$ in Remark 3.3).

Here, we provide a brief comparison with the following *straightforward randomization* of Algorithm 1, which randomly chooses between running the classic secretary problem without predictions and (roughly speaking) the greedy prediction-based procedure in Phase II in Algorithm 1. That is, given $\gamma \in [0, 1]$, with probability γ it runs the classic secretary problem, and with probability $1 - \gamma$, it runs the prediction-based algorithm that simply selects the first element with value greater or equal than $p^* - \lambda$ (if any). Note that its expected competitive ratio at least

$$\gamma \frac{1}{e} + (1 - \gamma)\left(\max\left\{1 - \frac{\lambda + \eta}{\mathrm{OPT}}, 0\right\}\right). \tag{1}$$

In order to compare Algorithm 1 with the *straightforward randomization*, we set $\gamma = 1/c$. This implies that both algorithms are at least $1/(ce)$-competitive in the worst-case when the predictions are poor. Having the same worst-case guarantee, we now focus on their performance in case when the prediction error η and the confidence parameter λ are small. In particular, let us consider the case where $\lambda + \eta = \delta \cdot \mathrm{OPT}$ for some small $\delta > 0$. Then, the expected competitive ratio in (1) reduces to

$$\frac{1}{ce} + \left(1 - \frac{1}{c}\right)(1 - \delta). \tag{2}$$

We now compare the expected competitive ratio of Algorithm 1 and its straightforward randomization, which read $f(c)(1 - \delta)$ and (2) respectively. In Fig. 2, we conduct a numerical experiment with fixed $\delta = 0.1$ and $\lambda + \eta = 0.1\mathrm{OPT}$. Our experimental data indicates that for $c \geq 1.185$, Algorithm 1 is at least $1/e$-competitive and it significantly outperforms the classic secretary algorithm as c increases. Furthermore, for $c \geq 1.605$ Algorithm 1 performs better than its straightforward randomization. On the other hand, we note that our experiments indicate that as δ increases the competitive advantage of Algorithm 1 over its straightforward randomization decreases.

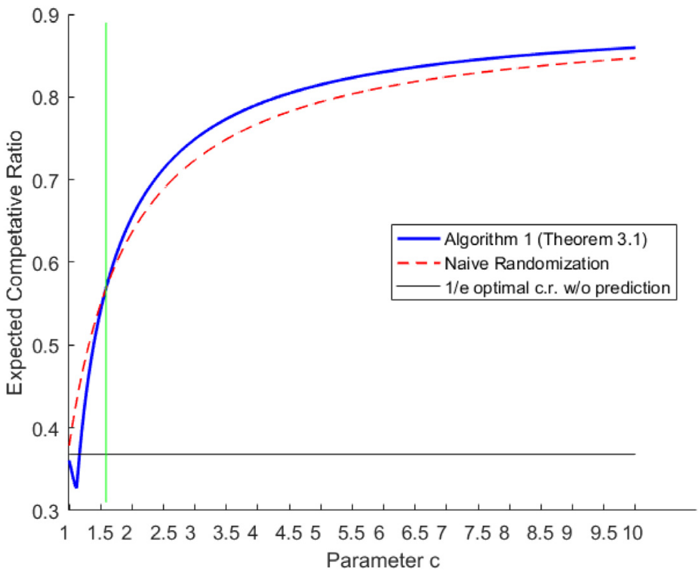

Fig. 2. The black horizontal line indicates the tight bound of $1/e$ for the classic secretary algorithm. The bold blue line is the performance guarantee for Algorithm 1; and the dashed red line is the performance guarantee for the obvious randomized algorithm. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

4. Online bipartite matching with random arrivals

In this section we consider a generalization of the value-maximization secretary problem discussed in Section 3. We study an online bipartite matching problem on a graph $G = (L \cup R, E)$ with edge weight function $w : E \to \mathbb{R}_{\geq 0}$. The vertices $\ell \in L$ arrive in a uniformly random order. Whenever a vertex $\ell \in L$ arrives, it reveals its neighbors $r \in R$ and what the corresponding edge weights $w(\ell, r)$ are. We then have the option to add an edge of the form (ℓ, r), provided r has not been matched in an earlier step. The goal is to select a set of edges, i.e., a matching, with maximum weight.

We assume that we are given, for all offline nodes $r \in R$, a prediction p_r^* for the value of the edge weight adjacent to r in some fixed optimal offline matching (which is zero if r is predicted not to be matched in this offline matching). That is, we predict that there exists some fixed optimal offline matching in which r is adjacent to an edge of weight p_r^* without predicting which particular edge this is. Note that the predictions $p = (p_1^*, \dots, p_r^*)$ implicitly provide a prediction for OPT, namely $\sum_{r \in R} p_r^*$. Recall that OPT was used as a prediction in Section 3. In the current setting, using OPT as a prediction does not seem to lead to valuable results. Roughly speaking, the problem is that OPT gives no information about how much the individual nodes in R contribute to OPT, which makes it hard to exploit it as a prediction.

It turns out that this type of predictions closely corresponds to the so-called *online vertex-weighted bipartite matching problem* where every offline node is given a weight w_r, and the goal is to select a matching with maximum weight, which is the sum of all weights w_r for which the corresponding r is matched in the online algorithm. This problem has both been studied under adversarial arrivals [17,43] and uniformly random arrivals [44,45]. In case the predictions are perfect, then, in order to find a matching with the corresponding predicted values, we just ignore all edges $w(\ell, r)$ that do not match the value p_r^*. This brings us in a special case of the online vertex-weighted bipartite matching problem.

The prediction error in this section will be defined as the maximum error over all predicted values and the minimum over all optimal matchings in G. We use $\mathcal{M}(G, w)$ to denote the set of all optimal matchings in G with respect to the weight function w, and then define

$$\eta = \min_{M \in \mathcal{M}(G,r)} \max_{r \in R} |p_r^* - w(M_r)|.$$

Here, we use $w(M_r)$ to denote the value of the edge adjacent to $r \in R$ in a given optimal solution with objective value $\mathrm{OPT} = \sum_r w(M_r)$.

In the next sections we will present deterministic and randomized algorithms, inspired by algorithms for the online vertex-weighted bipartite matching problem, that can be combined with the algorithm in [16] in order to obtain algorithms that incorporate the predictions. We start with a deterministic algorithm, which is the main result of this section.

4.1. Deterministic algorithm

We start by shortly recalling the algorithm by Kesselheim et al. [16] and the main ideas underlying its analysis. Similar to the algorithm for the classic secretary problem, it also consists of a sampling phase and a selection phase. During the sampling phase, $\lfloor n/e \rfloor$ many vertices are observed without adding any edges to the matching. For each subsequent vertex $\ell \in \{\lfloor n/e \rfloor + 1, \dots n\}$ that arrives an optimum matching for the currently revealed part of the graph is computed, and if (i) in that optimum matching ℓ is matched to vertex v, and (ii) v is currently unmatched for the algorithm, then edge (ℓ, v) is added to the matching. The most important feature in the analysis of the above algorithm is interpreting the random arrival order as a sequence of stochastically independent experiments. This allows them to lower bound the expected contribution to the objective function of every vertex ℓ in $\{\lfloor n/e \rfloor + 1, \dots n\}$ independently by

$$\frac{\lfloor n/e \rfloor}{\ell - 1} \cdot \frac{\mathrm{OPT}}{n}.$$

Their result that the algorithm is e-competitive then follows by adding up all of these expected contributions. Note that this is best possible, since the problem generalizes the classical secretary problem for which no better than e-competitive algorithm can exist.

As we will see, our algorithm is in part based on a modification of the algorithm of [16] that allows it to be combined with a third phase in which the predictions are exploited. This modification will require certain adaptations/generalizations to the analysis of the algorithm of [16] which we include for completeness in Appendix C.

We next give a simple deterministic greedy algorithm that provides a 1/2-approximation in the case when the predictions are perfect. It is very similar to a greedy algorithm given by Aggarwal et al. [17] for the online vertex-weighted bipartite matching problem. In fact, the 1/2-approximation from [17] is the best known deterministic algorithm for online vertex-weighted bipartite matching, to the best of our knowledge. Therefore, our aim is to give an algorithm that includes the predictions in such a way that, if the predictions are good (and we have high confidence in them), we should approach a 1/2-approximation, whereas if the predictions turn out to be poor, we are allowed to lose at most a constant factor w.r.t. the $1/e$-approximation in [16].

Although we do not emphasize it in the description, Algorithm 2 can be run in an online fashion. Provided that there exists an offline matching in which every $r \in R$ is adjacent to some edge with weight at least t_r, it can be shown, using the same arguments as given in [17], that Algorithm 2 yields a matching with weight at least $\frac{1}{2} \sum_r t_r$. We present the details in the proof of Theorem 4.1 later on.

Algorithm 3 is a deterministic algorithm that, similar to Algorithm 1, consists of three phases. The first two phases correspond to the two phases of the algorithm of Kesselheim et al. [16] as explained in the beginning of this section. In the third phase, whose purpose is to exploit the predictions, we then run the threshold greedy algorithm as described in Algorithm 2. Roughly speaking, we need to keep two things in mind in order to guarantee that we remain: constant-competitive in case the predictions are bad, and get an improved guarantee in case the predictions are good. In order to guarantee the latter, we should not match up too many offline nodes in the second phase, as this would block the possibility of selecting a good

ALGORITHM 2: Threshold greedy algorithm

Input : Thresholds $t = (t_1, \dots, t_{|R|})$ for offline nodes $r \in R$; ordered list $(v_1, \dots, v_\ell) \subseteq L$.
Output: Matching M

Set $M = \emptyset$.
for $i = 1, \dots, |L|$ **do**
  Set $r^i = \text{argmax}_r\{w(v_i, r) : r \in \mathcal{N}(v_i), w(v_i, r) \geq t_r$ and $r \notin R[M]\}$. /*Breaking ties arbitrarily
  if $r^i \neq \emptyset$ **then**
    Set $M = M \cup \{v_i, r^i\}$.
  end
end

solution in the third phase in case the predictions are good. On the other hand, in order to guarantee the former, we should not select to few offline nodes in the second phase, otherwise we are no longer guaranteed to be constant-competitive in case the predictions turn out to be poor. The analysis of Algorithm 3 given in Theorem 4.1 shows that it is possible to achieve both these properties.

For the sake of simplicity, in both the description of Algorithm 3 and its analysis in Theorem 4.1, we use a common λ to lower the predicted values (as we did in Section 3 for the secretary problem). Alternatively, one might investigate the use of a resource-specific value λ_r for this as well, but then the analysis seems to become much more involved.

ALGORITHM 3: Online bipartite matching algorithm with predictions

Input : Predictions $p^* = (p_1^*, \dots, p_{|R|}^*)$, confidence parameter $0 \leq \lambda \leq \min_r p_r^*$, and $c > d \geq 1$.
Output: Matching M.

Phase I: /*Algorithm from [16]
for $i = 1, \dots, \lfloor n/c \rfloor$ **do**
  Observe arrival of node ℓ_i, and store all the edges adjacent to it.
end
Let $L' = \{\ell_1, \dots, \ell_{\lfloor n/c \rfloor}\}$ and $M = \emptyset$.
Phase II:
for $i = \lfloor n/c \rfloor + 1, \dots, \lfloor n/d \rfloor$ **do**
  Set $L' = L' \cup \ell_i$.
  Set M^i = optimal matching on $G[L' \cup R]$.
  Let $e^i = (\ell_i, r)$ be the edge assigned to ℓ_i in M^i.
  if $M \cup e^i$ *is a matching* **then**
    Set $M = M \cup \{e^i\}$.
  end
end
Phase III: /*Threshold greedy algorithm
for $i = \lfloor n/d \rfloor + 1, \dots, n$ **do**
  Set $r^i = \text{argmax}_r\{w(v_i, r) : r \in \mathcal{N}(v_i), w(v_i, r) \geq p_r^* - \lambda$ and $r \notin R[M]\}$
  if $r^i \neq \emptyset$ **then**
    Set $M = M \cup \{\ell_i, r^i\}$.
  end
end

Theorem 4.1. *For any $\lambda \geq 0$ and $c > d \geq 1$, there is a deterministic algorithm for the online bipartite matching problem with uniformly random arrivals that is asymptotically $g_{c,d,\lambda}(\eta)$-competitive in expectation, where*

$$g_{c,d,\lambda}(\eta) = \left\{ \begin{array}{ll} \max\left\{\frac{1}{c}\ln(\frac{c}{d}), \left[\frac{d-1}{2c}\left(\max\left\{1 - \frac{(\lambda+\eta)|\psi|}{OPT}, 0\right\}\right)\right]\right\} & \text{if } 0 \leq \eta < \lambda, \\ \frac{1}{c}\ln(\frac{c}{d}) & \text{if } \eta \geq \lambda. \end{array} \right\},$$

and $|\psi|$ is the cardinality of an optimal (offline) matching ψ of the instance.

We note that both the cardinality $|\psi|$ and the value OPT of an optimal matching appear in the competitive ratio. If $|\psi|/\mathrm{OPT} \to \infty$ (when $\min\{|R|, |L|\} \to \infty$), this means that the weights of the edges in the optimal matching become very small, in which case the function $g_{c,d,\lambda}(\eta)$ in Theorem 4.1 boils down $\frac{1}{c}\ln(\frac{c}{d})$, which does not give an improved guarantee in case the predictions are accurate. Roughly speaking, Theorem 4.1 is only of interest when $|\psi|/\mathrm{OPT}$ stays bounded away from zero.

Note that Theorem 4.1 tells us that we will always have a worst-case guarantee of $\frac{1}{c}\ln(\frac{c}{d})$ which is constant because c and d are constant. In order to see how we can get close to a 1/2-approximation in case the predictions are close to being perfect, we should set λ to be small. If then $\eta < \lambda$, roughly speaking meaning that the prediction is close to perfect, we get a bound of $(d-1)/2c$. If we therefore choose c/d close to 1 (but still constant to ensure a constant worst-case guarantee), we can get arbitrarily close to a 1/2-approximation.

Proof of Theorem 4.1. We provide two lower bounds on the expected value of the matching M output by Algorithm 3.

First of all, the analysis of the algorithm of Kesselheim et al. [16] can be generalized to the setting we consider in the first and second phase. In particular, their algorithm then yields a

$$\left(\frac{1}{c} - \frac{1}{n}\right)\ln\left(\frac{c}{d}\right)\text{-competitive approximation.}$$

(The $1/n$ factor is hidden in the statement of Theorem 4.1, following Remark 1.2, for sake of readability.) For completeness, we present a proof of this statement in Appendix C.

The second bound we prove on the expected value of the matching M is based on the threshold greedy algorithm we use in the third phase. Let $\psi \in \mathcal{M}(G, w)$ be an optimal (offline) matching, with objective value OPT, and suppose that

$$\eta = \max_{r\in R} |\psi_r - p_r^*| < \lambda. \tag{3}$$

The proof of the algorithm in [16] analyzes the expected value of the online vertices in L. Here we take a different approach and study the expected value of the edge weights adjacent to the nodes in $r \in R$. Fix some $r \in R$ and consider the edge (ℓ, r) that is matched to r in the optimal offline matching ψ (if any).

Let X_r be a random variable denoting the value of node $r \in R$ in the online matching M chosen by Algorithm 3. Let Y_ℓ be a random variable that denotes the value of node $\ell \in L[\psi]$ in the online matching M. It is not hard to see that

$$\mathbb{E}(w(M)) = \sum_{r\in R}\mathbb{E}(X_r) \quad \text{and} \quad \mathbb{E}(w(M)) \geq \sum_{\ell\in L[\psi]}\mathbb{E}(Y_\ell). \tag{4}$$

For the inequality, note that for any fixed permutation the value of the obtained matching is always larger or equal to the sum of the values that were matched to the nodes $\ell \in L[\psi]$.

Now, consider a fixed edge (ℓ, r) that is contained in ψ. We will lower bound the expectation $\mathbb{E}(X_r + Y_\ell)$ based on the expected value these nodes would receive, roughly speaking, if they get matched in the third phase. Therefore, suppose for now that the event that r did not get matched in the second phase occurs, as well as the event that ℓ appears in the part of the uniformly random input sequence/permutation considered in the third phase. We will later lower bound the probability with which these events occur. By definition of the greedy threshold algorithm, we know that at the end of Phase III either node $r \in R$ is matched, or otherwise at least node ℓ is matched to some other $r' \in R$ for which

$$w(\ell, r') \geq w(\ell, r) \geq p_r^* - \eta \geq p_r^* - \lambda. \tag{5}$$

To see this, consider the following three cases: either ℓ got matched to r, or it got matched to some other r' for which $w(\ell, r') \geq w(\ell, r) \geq p_r^* - \lambda$, or r was matched earlier during the third phase to some other ℓ' for which $w(\ell', r) \geq p_r^* - \lambda$.

Looking closely at the analysis of Kesselheim et al. [16], see Appendix C.3, it follows that the probability that a fixed node r did not get matched in the second phase satisfies

$$\mathrm{P}(r \text{ was not matched in Phase II}) \geq \frac{d}{c} - o(1).$$

This lower bound is true, independently of whether or not ℓ appeared in Phase III, or the first two phases (see Appendix C). This event that ℓ appeared in Phase III holds with probability $(1 - 1/d)$. Therefore,

$$\mathrm{P}(r \text{ was not matched in Phase II } \textit{and } \ell \text{ arrives in Phase III}) \geq \left(\frac{d}{c} - o(1)\right)\left(1 - \frac{1}{d}\right) = \frac{d-1}{c} - o(1).$$

Furthermore, we have from the arguments before (5) that under this condition either $X_r \geq p_r^* - \lambda$ or $Y_\ell \geq p_r^* - \lambda$. This implies that

$$\mathbb{E}(X_r + Y_\ell) \geq \left(\frac{d-1}{c} - o(1)\right)(p_r^* - \lambda). \tag{6}$$

By adding up the (in)equalities in (4), and combining this with (6), we get

$$2 \cdot \mathbb{E}(w(M)) \geq \left(\frac{d-1}{c} - o(1)\right) \sum_{r \in R[\psi]} (p_r^* - \lambda) \geq \left(\frac{d-1}{c} - o(1)\right)(\mathrm{OPT} - (\lambda + \eta)|\psi|)$$

assuming that $\mathrm{OPT} - (\lambda + \eta)|\psi| \geq 0$. In the last inequality, we use the definition of η in (3). Rewriting this gives

$$\mathbb{E}(w(M)) \geq \left(\frac{d-1}{2c} - o(1)\right)\left(1 - \frac{(\lambda + \eta)|\psi|}{\mathrm{OPT}}\right) \cdot \mathrm{OPT},$$

which yields the desired bound. □

It is interesting to note that Theorem 4.1 does not seem to hold if we interchange the second and the third phase. In particular, if the predictions are too low, we most likely match up too many nodes in $r \in R$ already in the second phase (which is now the threshold greedy algorithm). Indeed, a similar calculation as in the proof of Theorem 4.1 seems to yield a worst bound, but we leave this to the interested reader.

4.2. Randomized algorithm

If we allow randomization, we can give better approximation guarantees than the algorithm given in the previous section, by using a convex combination of the algorithm of Kesselheim et al. [16], and the randomized algorithm of Huang et al. [45] for online vertex-weighted bipartite matching with uniformly random arrivals. We only sketch this idea here. We give a simple, generic way to reduce an instance of online bipartite matching with predictions p_r^* for $r \in R$ to an instance of online vertex-weighted bipartite matching with vertex weights (that applies in case the predictions are accurate).

Suppose we are given an algorithm $\mathcal{A}$ for instances of the online vertex-weighted bipartite matching problem that is α-competitive. If the predictions would be perfect, we can do the following. Whenever a node arrives, we only focus on edges whose weight matches the predicted value p_r^*. To be precise, consider the instance in which we leave out all edges (ℓ, r) for which $w(\ell, r) \neq p_r^*$. If we run algorithm $\mathcal{A}$ on this instance, we obtain a matching that is α-competitive with respect to the optimal matching under the predicted values. Since the predictions are perfect, this also gives an α-competitive algorithm for the original instance (on which we run the algorithm that makes the same decisions as algorithm $\mathcal{A}$).

In case the predictions are accurate, but not perfect, we need a slightly more general version of this idea. Fix some small parameter $\lambda > 0$ up front and assume that $\eta < \lambda$. We assume λ and η to be small as we are arguing about the case where the predictions are accurate. Whenever a vertex ℓ arrives online we only take into account edges (ℓ, r) with the property that $w(\ell, r) \in [p_r^* - \lambda, p_r^* + \lambda]$, and ignore all edges that do not satisfy this property. To be precise, we can consider the instance of the vertex-weighted bipartite matching problem in which we leave out all edges for which $w(\ell, r) \in [p_r^* - \lambda, p_r^* + \lambda]$ and where the (offline) vertices are given the weights p_r^*. The decision we make in the original instance, are the decisions that $\mathcal{A}$ would make on the modified instance. Because we assume η to be small, this will give us an algorithm that is approximately $\beta(\lambda)$-competitive, with $\lim_{\lambda \to 0} \beta(\lambda) = \alpha$.

The 0.6534-competitive algorithm of Huang et al. [45] is the currently best known randomized algorithm $\mathcal{A}$ for online vertex-weighted bipartite matching with uniformly random arrivals, and can be used for our purposes with $\alpha = 0.6534$.

5. Deterministic graphic matroid secretary algorithm

In this section we will study the graphic matroid secretary problem. Here, we are given a (connected) graph $G = (V, E)$ of which the (uniform random) input sequence is formed by the edges of E, i.e., the edges in E arrive one-by-one in an online fashion. There is an edge weight function $w : 2^E \to \mathbb{R}_{\geq 0}$ and a weight is revealed if an edge arrives. The goal is to select a forest (i.e., a subset of edges that does not give rise to a cycle) of maximum weight. The possible forests of G form the *independent sets of the graphical matroid on* G. It is well-known that the offline optimal solution of this problem is maximum weight spanning tree. It can be found by the greedy algorithm that first orders all the edge weights in decreasing order. It then builds a spanning tree by repeatedly adding edges to an initially empty set, all the while making sure that no cycles are introduced. This is known as Kruskal's algorithm.

We will next explain the predictions that we consider in this section. For every node $v \in V$, we let p_v^* be a prediction for the *maximum edge weight* $\max_{u \in \mathcal{N}(v)} w_{uv}$ adjacent to $v \in V$. The prediction error is defined by

$$\eta = \max_{v \in V} \left| p_v^* - w_{\max}(v) \right|, \quad \text{where} \quad w_{\max}(v) = \max_{u \in \mathcal{N}(v)} w_{uv}.$$

Remark 5.1. Although the given prediction is formulated independently of any optimal solution (as we did in the previous sections), it is nevertheless equivalent to a prediction regarding the maximum weight w_{uv} adjacent to $v \in V$ in an optimal (greedy) solution. To see this, note that the first time Kruskal's algorithm encounters an edge weight w_{uv} adjacent to v, for some u, it can always be added as currently there is no edge adjacent to v. I.e., adding the edge $\{u, v\}$ cannot create a cycle.

Before we give the main result of this section, we first provide a *deterministic* $(1/4 - o(1))$-competitive algorithm for the graphic matroid secretary problem in Section 5.1, which is of independent interest. Although this does not improve over the approximation guarantee of the randomized algorithm of Soto et al. [18], our algorithm is deterministic. We then continue with an algorithm incorporating the predictions in Section 5.2.

5.1. Deterministic approximation algorithm

In this section, we provide a deterministic $(1/4 - o(1))$-approximation for the graphic matroid secretary problem. For a given undirected graph $G = (V, E)$, we use the bipartite graph interpretation that was also used in [19]. That is, we consider the bipartite graph $B_G = (E \cup V, A)$, where an edge $\{e, v\} \in A$, for $e \in E$ and $v \in V$, if and only if $v \in e$. Note that this means that every $e = \{u, v\}$ is adjacent to precisely u and v in the bipartite graph B_G. Moreover, the edge weights for $\{e, v\}$ and $\{e, u\}$ are both w_e (which is revealed upon arrival of the element e).[8] We emphasize that in this setting, the $e \in E$ are the elements that arrive online.

Algorithm 4 is very similar to the algorithm in [16] with the only difference that we allow an edge $\{e, u\}$ or $\{e, v\}$ to be added to the currently selected matching M in B_G if and only if *both* nodes u and v are currently not matched in M. In this section we often represent a (partial) matching in B_G by a directed graph (of which its undirected counterpart does not contain any cycle). In particular, given some matching M in B_G, we consider the directed graph D_M with node set V. There is a directed edge (u, v) if and only if $\{e, v\}$ is an edge in M, where $e = \{u, v\} \in E$. Note that every node in D_M has an in-degree of at most one as M is a matching.

Using the graph D_M it follows that if both u and v are not matched in the current matching M, then adding the edge $\{e, u\}$ or $\{e, v\}$ can never create a cycle in the graph formed by the elements $e \in E$ matched up by M, called $E[M]$, which is the currently chosen independent set in the graphic matroid. We will next prove this claim. Note that the set $E[M]$ is precisely the undirected counterpart of the edges in D_M together with $\{u, v\}$. For sake of contradiction, suppose adding the edge $\{u, v\}$ to $E[M]$ would create an (undirected) cycle C. As both u and v have in-degree zero (as they are unmatched in M), it follows that some node on the cycle C must have two incoming directed edges in the graph D_M. This yields a contradiction.

We note that, although u and v being unmatched is sufficient to guarantee that the edge $\{u, v\}$ does not create a cycle, this is by no means a necessary condition.

Although the optimal choice of the parameter c equals $c = 2$ in Algorithm 4, we include a general description which is handy for when we want to include predictions in the next section.

ALGORITHM 4: Deterministic graphic matroid secretary algorithm

Input : Bipartite graph $G_B = (E \cup V, A)$ for undirected weighted graph $G = (V, E)$, with $|E| = m$, and parameter $c > 0$.
Output: Matching M of G_B corresponding to forest in G.

Phase I:
for $i = 1, \dots, \lfloor m/c \rfloor$ **do**
  Observe arrival of element e_i, but do nothing.
end
Let $E' = \{e_1, \dots, e_{\lfloor m/c \rfloor}\}$ and $M = \emptyset$.
Phase II:
for $i = \lfloor m/c \rfloor + 1, \dots, m$ **do**
  Let $E' = E' \cup e_i$.
  Let M^i = optimal matching on $B_G[E' \cup V]$.
  Let $a^i = \{e_i, u\}$ be the edge assigned to $e_i = \{u, v\}$ in M^i (if any).
  if *$M \cup a^i$ is a matching and both u and v are unmatched in M* **then**
    Set $M = M \cup a^i$.
  end
end

Theorem 5.2. *Algorithm 4 is a deterministic $(1/4 - o(1))$-competitive algorithm for the graphic matroid secretary problem for $c = 2$.*

Proof (*Sketch*)**.** The proof proceeds along similar lines as the proof in Appendix C, but is technically more involved this time. Roughly speaking, it can be shown that the expected contribution of node ℓ in Phase II of Algorithm 4 this time is at least

$$\frac{\lfloor m/c \rfloor - 1}{(\ell - 1) - 1} \cdot \frac{\lfloor m/c \rfloor}{\ell - 1} \cdot \frac{OPT}{m},$$

[8] We call the edges in E (of the original graph G) elements, in order to avoid confusion with the edges of B_G.

ALGORITHM 5: Graphic matroid secretary algorithm with predictions

Input : Bipartite graph $G_B = (E \cup V, A)$ for undirected graph $G = (V, E)$ with $|E| = m$. Predictions $p = (p_1^*, \ldots, p_n^*)$. Confidence parameter $0 \leq \lambda \leq \min_i p_i^*$ and $c > d \geq 1$.
Output: Matching M of G_B corresponding to forest in G.

Phase I:
for $i = 1, \ldots, \lfloor m/c \rfloor$ **do**
  Let $e_i = \{u, v\}$.
  Set $t_v = \max\{t_v, w(u, v)\}$ and $t_u = \max\{t_u, w(u, v)\}$.
end
Let $E' = \{e_1, \ldots, e_{\lfloor m/c \rfloor}\}$ and $M = \emptyset$.
Phase II:
for $i = \lfloor m/c \rfloor + 1, \ldots, \lfloor m/d \rfloor$ **do**
  Let $e_i = \{u, v\}$, $S = \{x \in \{u, v\} : x \notin E[M]$ and $w(u, v) \geq \max\{t_x, p_x^* - \lambda\}\}$ and $y_i = \mathrm{argmax}_{x \in S} p_x^* - \lambda$.
  if $E[M] \cup \{e_i\}$ *does not contain a cycle* **then**
    Set $M = M \cup \{e_i, y_i\}$.
  end
end
Phase III:
for $i = \lfloor m/d \rfloor + 1, \ldots, m$ **do**
  Let $E' = E' \cup e_i$.
  Let M^i = optimal matching on $B_G[E' \cup V]$.
  Let $a^i = \{e_i, u\}$ be the edge assigned to $e_i = \{u, v\}$ in M^i (if any).
  if $M \cup a^i$ *is a matching and both u and v are unmatched in M* **then**
    Set $M = M \cup a^i$.
  end
end

where $m = |E|$ is the total number of edges and c a parameter that determines the length of the two phases in Algorithm 4. Summing up over all ℓ in Phase II, we obtain that the expected value obtained by Algorithm 4 is at least

$$\frac{OPT}{m} \sum_{\ell=\lfloor m/c \rfloor+1}^{m} \frac{\lfloor m/c \rfloor - 1}{(\ell - 1) - 1} \cdot \frac{\lfloor m/c \rfloor}{\ell - 1} \geq \mathrm{OPT}\left(\frac{c-1}{c^2} - o(1)\right).$$

This quantity is maximized for $c = 2$, yielding the desired result. The full proof is given in Appendix D □

5.2. Algorithm including predictions

In this section we will augment Algorithm 4 with the predictions for the maximum edge weights adjacent to the nodes in V. We will use the bipartite graph representation B_G as introduced in Section 5. Algorithm 5 consists of three phases, similar to Algorithm 3.

Instead of exploiting the predictions in Phase III, we already exploit them in Phase II for technical reasons. Roughly speaking, in Phase II, we run a greedy-like algorithm that selects for every node $v \in V$ at most one edge that satisfies a threshold based on the prediction for node v and the best edges seen in Phase I adjacent to v. The latter is done to guarantee that we do not select too many edges when the predictions are poor (in particular when they are too low). A similar idea has already been used in Algorithm 1 for the secretary problem.

Theorem 5.3. *For any $\lambda \geq 0$ and $c > d \geq 1$, there is a deterministic algorithm for the graphic matroid secretary problem that is asymptotically $g_{c,d,\lambda}(\eta)$-competitive in expectation, where*

$$g_{c,d,\lambda}(\eta) = \begin{cases} \max\left\{\frac{d-1}{c^2}, \frac{1}{2}\left(\frac{1}{d}-\frac{1}{c}\right)\left(1-\frac{2(\lambda+\eta)|V|}{OPT}\right)\right\} & \text{if } 0 \leq \eta < \lambda, \\ \frac{d-1}{c^2} & \text{if } \eta \geq \lambda. \end{cases}$$

If λ is small, we roughly obtain a bound of $(1/d-1/c)/2$ in case η is small, and a bound of $(d-1)/c^2$ if η is large. Note that the roles of c and d have interchanged w.r.t. Algorithm 3 as we now exploit the predictions in Phase II instead of Phase III. Roughly speaking, if $d \to 1$ and $c \to \infty$ we approach a bound of $1/2$ if the predictions are good, whereas the bound of $(d-1)/c^2$ becomes arbitrarily bad.

Proof of Theorem 5.3. As in the proof of Theorem 4.1, we provide two lower bounds on the expected value on the matching M outputted by Algorithm 5. We first provide a bound of

$$\frac{1}{2}\left(\frac{1}{d}-\frac{1}{c}\right)\left(1-\frac{2(\lambda+\eta)|V|}{\mathrm{OPT}}\right)\mathrm{OPT}$$

in case the prediction error is small, i.e., when $\eta < \lambda$. For simplicity we focus on the case where for each $v \in V$, the weights of the edges adjacent to v in G are distinct.[9]

For every $v \in V$, let $e_{max}(v)$ be the (unique) edge adjacent to v with maximum weight among all edges adjacent to v. Consider the fixed order $(e_1, \dots, e_m)$ in which the elements in E arrive online, and define $Q = \{v \in V : e_{max}(v) \text{ arrives in Phase II}\}$. We will show that the total weight of all edges selected in Phase II is at least $\frac{1}{2}\sum_{v \in Q}(p_v^* - (\lambda+\eta))$. Let $T \subseteq Q$ be the set of nodes for which the edge $e_{\max}(v)$ arrives in Phase II, but for which v does not get matched up in Phase II.

In particular, let $v \in T$ and consider the step ℓ in Phase II in which $e_{\max}(v) = \{u, v\}$ arrived. By definition of η, and because $\eta < \lambda$, we have

$$w(u,v) = w_{\max}(v) \geq \max\{t_v, p_v^* - \eta\} \geq \max\{t_v, p_v^* - \lambda\}, \tag{7}$$

and so the pair $\{e_i, v\}$ is eligible (in the sense that $v \in S$). Since v did not get matched, one of the following two holds:

(i) The edge $e_{\max}(v)$ got matched up with u.
(ii) Adding the edge $\{e_{\max}(v), v\}$ to M would have yielded a cycle in $E[M] \cup e_{\max}(v)$.

Note that it can never be the case that we do *not* match up $e_{\max}(v)$ to u for the reason that it would create a cycle. This is impossible as both u and v are unmatched.

Now, in the first case, since u is matched it must hold that $w(u,v) \geq \max\{t_u, p_u^* - \lambda\}$, and $p_u^* - \lambda \geq p_v^* - \lambda$ as v was eligible to be matched up in the online matching M (but it did not happen). Further, combining (7) and the definition of y_i in Phase II, yields

$$2w(u,v) \geq (p_u^* - \lambda) + (p_v^* - \eta) \geq (p_u^* - \eta - \lambda) + (p_v^* - \eta - \lambda). \tag{8}$$

We call u the *(i)-proxy* of v in this case.

In the second case, if adding $e_{\max}(v)$ would have created an (undirected) cycle in the set of elements (i.e., the forest) selected so far, this yields a unique directed cycle in the graph D_M defined in the previous section. If not, then there would be a node with two incoming arcs in D_M, as every arc on the cycle is oriented in some direction. This would imply that M is not a matching.

[9] For the general case, one can use a global ordering on all edges in E and break ties where needed.

Let $e' = \{u, z\} \in E$ be the element corresponding to the incoming arc at u in D_M. Note that by assumption u is already matched up, as $e_{\max(v)}$ creates a directed cycle in $D_{M\cup e_{\max}(v)}$. That is, we have $\{e', u\} \in M$. Then, by definition of η, we have

$$\eta + p_u^* \geq e_{\max}(u) \geq w(u, v) \geq p_v^* - \eta, \tag{9}$$

where the last inequality holds by (7). Combining (9) with the fact that $w(u, z) \geq p_u^* - \lambda$ (because $\{u, z\}$ got matched up to u), and the fact that $e_{\max}(v) \geq p_v^* - \eta$, by (7), it follows that

$$2w(u, z) \geq [p_u^* - (\lambda + \eta)] + [p_v^* - (\lambda + \eta)]. \tag{10}$$

In this case, we call u the *(ii)-proxy* of v.

Claim 5.4. *For any distinct $v, v' \in T$, their corresponding proxies u and u' are also distinct.*

Proof. Suppose that $u = u'$. The proof proceeds by case distinction based on the proxy types.

1. $u = u'$ is defined as (i)-proxy for both v and v': This cannot happen as $u = u'$ would then have been matched up twice by Algorithm 5.
2. $u = u'$ is defined as (ii)-proxy for both v and v': In this case there is a directed cycle with the arc $(u, v) = (u', v)$ and another directed cycle with the arc $(u', v') = (u, v')$. Hence, there is a vertex with two incoming arcs in D_M. This also means that Algorithm 5 has matched up a vertex twice, which is not possible.
3. $u = u'$ is defined as (i)-proxy for v and as (ii)-proxy for v': Then $e_{\max}(v)$, which gets matched up with $u = u'$, must have arrived before $e_{\max}(v')$. If not, then both v' and $u = u'$ would have been unmatched when $e_{\max}(v')$ arrived and we could have matched it up with at least v' (as this cannot create a cycle since $u = u'$ is also not matched at that time). This means that when $e_{\max}(v')$ arrived, the reason that we did not match it up to v' is because this would create a directed cycle in $D_{M\cup e_{\max}(v')}$. But, as u has an incoming arc from v in D_M, this means that the directed cycle goes through v, which implies that v *did* get matched up in Phase II, which we assumed was not the case.

This concludes the proof of the claim. □

Using Claim 5.4 in combination with (8) and (10), we then find that

$$w[M_{II}] \geq \frac{1}{2} \sum_{v\in Q} [p_v^* - (\lambda + \eta)],$$

where M_{II} contains all the edges obtained in Phase II. Roughly speaking, for every edge $e_{\max}(v)$ that we cannot select in Phase II, there is some other edge selected in Phase II that 'covers' its weight in the summation above (and for every such v we can find a unique edge that has this property).

Now, in general, we have a uniformly random arrival order, and therefore, for every $v \in V$, the probability that edge $e_{\max}(v)$ arrives in Phase II equals $\frac{1}{d} - \frac{1}{c}$. Therefore, with expectation taken over the arrival order, we have

$$\mathbb{E}[M_{II}] \geq \frac{1}{2}\left(\frac{1}{d} - \frac{1}{c}\right) \sum_{v\in V} (p_v^* - (\lambda + \eta)) \geq \frac{1}{2}\left(\frac{1}{d} - \frac{1}{c}\right)\left(1 - \frac{2(\lambda + \eta)|V|}{\text{OPT}}\right) \text{OPT}.$$

We continue with the worst-case bound that holds even if the prediction error is large. We first analyze the probability that two given nodes u and v do not get matched up in Phase II. Here, we will use the thresholds t_v defined in Algorithm 5.

Conditioned on the set of elements A that arrived in Phase I/II, the probability that the maximum edge weight adjacent to v, over all edges adjacent to v in A, appears in Phase I is at equal to (d/c). This implies that v will not get matched up in Phase II, by definition of Algorithm 5. The worst-case bound of $(d-1)/c^2$ is proven in Appendix E along similar lines as Appendices C and D. □

5.3. Randomized algorithm with predictions

For the graphic matroid secretary problem with prediction there is also a (randomized) convex combination possible between two algorithms that either fully exploit, or completely ignore, the prediction, respectively. We can use the 4-competitive algorithm of Soto et al. [18] as the algorithm that completely ignores the prediction. An algorithm that fully exploits the predictions is given by the one that runs Phase II of Algorithm 5 on the whole input sequence. That is, we lower every prediction p_x^* by λ. We then repeatedly add an edge whose weight exceeds $p_x^* - \lambda$ whenever it does not create a cycle with the already selected edges. In case the predictions are close to perfect, this algorithm is 1/2-competitive. There might be better randomized (or deterministic) algorithms for exploiting the predictions when they are accurate, but we leave this for future work.

6. Conclusion

Our results can be seen as the first evidence that online selection problems are a promising area for the incorporation of machine learned advice following the frameworks of [5,6]. Many interesting problems and directions remain open. For example, does there exist a natural prediction model for the general matroid secretary problem? It is still open whether this problem admits a constant-competitive algorithm. Is it possible to show that there exists an algorithm under a natural prediction model that is constant-competitive for accurate predictions, and that is still $O(1/\log(\log(r)))$-competitive in the worst case, matching the results in [23,24]? Furthermore, although our results are optimal within this specific three phased approach, it remains an open question whether they are optimal in general for the respective problems.

Data availability

No data was used for the research described in the article

Acknowledgments

An extended abstract of this work appeared in the Proceedings of the 34th Conference on Neural Information Processing Systems (NeurIPS 2020). This version contains all proofs and additional remarks that were omitted in the NeurIPS proceedings version (due to space constraints). We would like to thank Kurt Mehlhorn for drawing our attention to the area of algorithms augmented with machine learned predictions. Antonios Antoniadis was supported by German Science Foundation grant AN 1262/1-1.

Appendix A. Smoothing out the discontinuity of Theorem 3.1

We show that the discontinuity at $\eta = \lambda$, in Theorem 3.1, can be smoothed out by selecting λ according to some distribution with mean representing our confidence in the prediction p^*. Further, we study the competitive ratio as a function of the prediction error η.

We note that Algorithm 1 although deterministic, can be relatively easily transformed to an algorithm that picks the confidence parameter $\lambda \in [0, p^*]$ according to some probability distribution. Algorithm 1 is then the special case where the whole probability mass of the distribution is at one point in $[0, p^*]$.

This naturally gives rise to the question of whether there exists a distribution that outperforms the deterministic algorithm. It can be relatively easily seen, that the deterministic algorithm with $\lambda = \eta$ is the best possible competitive ratio that can be obtained with such an approach. Therefore, a randomized algorithm can at best match the deterministic one with $\lambda = \eta$, and this happens only in the case in the

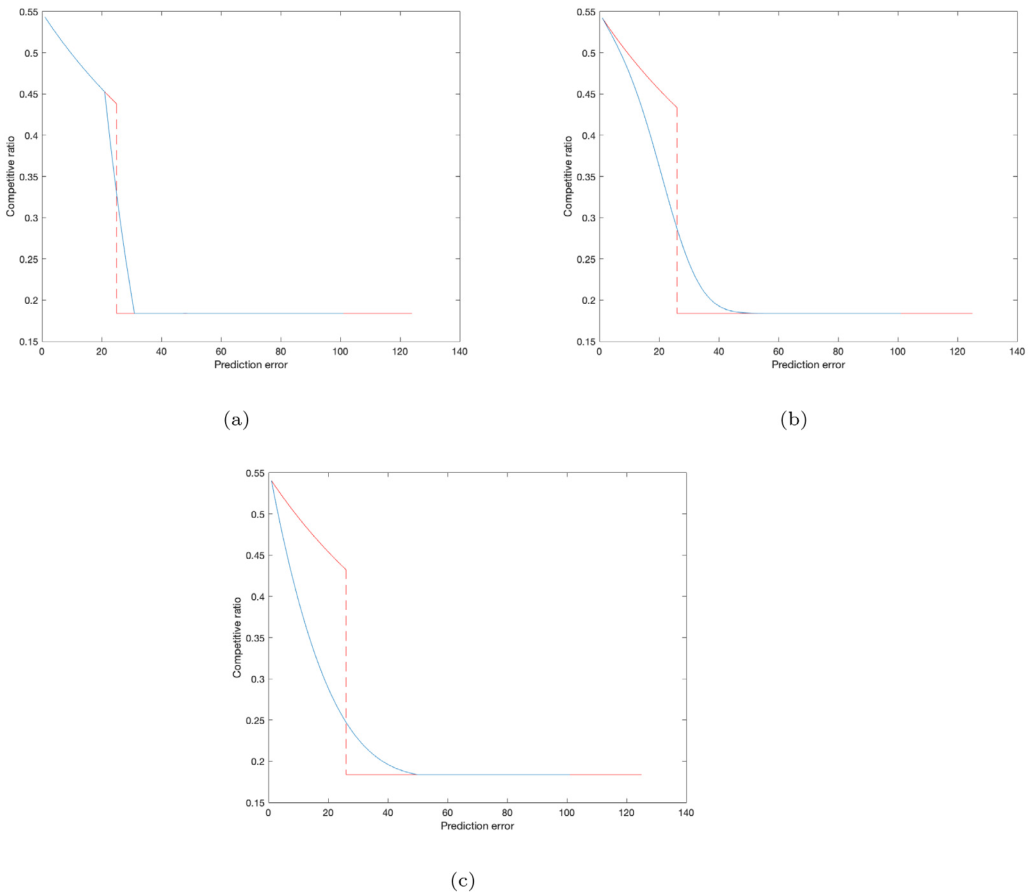

Fig. A.3. A comparison the deterministic (in red) and randomized (in blue) choice of λ. The y-axes are the competitive ratio, the x-axes are the prediction error η, and all figures consider $p^* = 100$. In Fig. A.3(a) we take $\lambda = 25$ for the deterministic algorithm and choose λ according to the uniform distribution in $(20, 30)$ for the randomized one. In Fig. A.3(b) we have $\lambda \approx 25$ for the deterministic one and the normal distribution with mean 25 and variance 10. Finally, in Fig. A.3(c) we have $\lambda \approx 25$, and the normal distribution with mean 0 and variance 32. All plots have $c = 2$. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

center of mass of the used distribution is at η. Since η is unknown to the algorithm, it is not possible to select a distribution that outperforms any deterministic algorithm for all possible η's.

Despite that, it may still be advantageous to pick λ at random according to some distribution, in order to avoid the "jump" that occurs at $\eta = \lambda$ in the competitive ratio of the deterministic algorithm. In particular for an appropriate distribution with density function $h_\lambda(x)$ the expected competitive ratio is given by:

$$\mathbb{E}_\lambda \left[g_{c,\lambda}(\eta)\right] = Pr[\lambda < \eta] \cdot \frac{1}{ce} + f(c) \int_\eta^{p^*} h_\lambda(x) \left(1 - \frac{x + \eta}{OPT}\right) dx,$$

which can be seen as a convex combination of two competitive ratios.

Some example distributions and how they compare to an algorithm that selects λ deterministically can be seen in Fig. A.3.

Appendix B. Perfect matching instances

Given an undirected weighted bipartite graph $G' = (L' \cup R', E', w')$ we construct an augmented bipartite graph $G = (L \cup R, E, w)$ as follows:

(i) the left node set $L = L'$; (ii) the right node set $R = R' \cup L'$; (iii) the edge set $E = E' \cup F'$ where the set F' consists of edges $\{u_i, v_i\}$ such that u_i and v_i are the ith node in L and L' respectively, for all $i \in \{1, \dots, |L|\}$; (iv) $w(e) = w'(e)$ for all edges $e \in E'$ and $w(e) = 0$ for all edges $e \in F'$.

We call the resulting bipartite graph G *perfect*.

Fact B.1. *Suppose $G = (L \cup R, E, w)$ is a perfect bipartite graph. Let $\ell \in \{1, \dots, |L|\}$ be an arbitrary index and $\mathcal{L}(\ell) = \{S \subseteq L \ : \ |S| = \ell\}$ be the set of all subsets of nodes in L of size ℓ. Then, for every subset $S \in \mathcal{L}(\ell)$ the induced subgraph $G[S \cup N(S)]$ has a perfect matching M_S of size ℓ, i.e., $|M_S| = |S| = \ell$.*

Appendix C. General analysis of the algorithm of Kesselheim et al. [16]

In this section, we analyze a modified version of the algorithm of Kesselheim et al. [16], see Algorithm 6. Our analysis extends the proof techniques presented in [16, Lemma 1].

Theorem C.1. *Given a perfect bipartite graph, Algorithm 6 is $(\frac{1}{c} - \frac{1}{n}) \ln \frac{c}{d}$ competitive in expectation. In addition, the expected weighted contribution of the nodes $\{\lfloor n/d \rfloor + 1, \dots, n\}$ to the online matching M is $OPT \cdot (\frac{1}{c} - \frac{1}{n}) \ln d$.*

For convenience of notation, we will number the vertices in L from 1 to n in the random order they are presented to the algorithm. Hence, we will use the variable ℓ as an integer, the name of an iteration and the name of the current node (the last so far).

ALGORITHM 6: Online bipartite matching algorithm (under uniformly random vertex arrivals)

Input : Vertex set R and cardinality $|L| = n$.
Output: Matching M.

Phase I:
for $\ell = 1, \dots, \lfloor n/c \rfloor$ **do**
  Observe arrival of node ℓ, but do nothing.
end
Let $L' = \{1, \dots, \lfloor n/c \rfloor\}$ and $M = \emptyset$.
Phase II:
for $\ell = \lfloor n/c \rfloor + 1, \dots, \lfloor n/d \rfloor$ **do**
  Let $L' = L' \cup \ell$.
  Let $M^{(\ell)}$ = optimal matching on $G[L' \cup R]$.
  Let $e^{(\ell)} = (\ell, r)$ be the edge assigned to ℓ in $M^{(\ell)}$.
  if $M \cup e^{(\ell)}$ *is a matching* **then**
    Set $M = M \cup \{e^{(\ell)}\}$.
  end
end

Organization

In Appendix C.1, we present the notation. In Appendix C.2, we give the main structural result and prove Theorem C.1. In addition, in Appendix C.3, we give a lower bound on the probability that an arbitrary node $r \in R$ remains unmatched after the completion of Phase II.

C.1. Notation

Consider the following random process:

Sample uniformly at random a permutation of the nodes L. Let L_ℓ be a list containing the first ℓ nodes in L, in the order as they appear, and let $M^{(\ell)}$ be the corresponding optimum matching of the induced subgraph $G^{(\ell)}$ on the node set $L_\ell \cup N(L_\ell)$.

Let $E^{(\ell)}$ be the event $\{e^{(\ell)} \cup M$ is a matching$\}$, where (r.v.) M is the current online matching. Note that the existence of edge (r.v.) $e^{(\ell)}$ is guaranteed by Fact B.1 and G is a perfect bipartite graph. We define a random variable

$$A_\ell = \begin{cases} w(e^{(\ell)}) & \text{, if event } E^{(\ell)} \text{ occur;} \\ 0 & \text{, otherwise.} \end{cases}$$

C.2. Structural Lemma

Lemma C.2. *Suppose $G = (L \cup R, E, w)$ is a perfect bipartite graph. Then, for every $c > 1$ it holds for every $\ell \in \{\lfloor n/c \rfloor + 1, \dots, n\}$ that*

$$\mathbb{E}[A_\ell] \geq \frac{\lfloor n/c \rfloor}{n} \cdot \frac{OPT}{\ell - 1}.$$

Before we prove Lemma C.2, we show that it implies Theorem C.1.

C.2.1. Proof of Theorem C.1

Using Lemma C.2, we have

$$\mathbb{E}\left[\sum_{\ell=1}^{n/d} A_\ell\right] = \sum_{\ell=\lfloor n/c \rfloor+1}^{n/d} \mathbb{E}[A_\ell] \geq \sum_{\ell=\lfloor n/c \rfloor+1}^{n/d} \frac{\lfloor n/c \rfloor}{n} \cdot \frac{OPT}{\ell - 1} \geq OPT \cdot \left(\frac{1}{c} - \frac{1}{n}\right) \cdot \ln \frac{c}{d},$$

where the inequalities follow by combining $\frac{\lfloor n/c \rfloor}{n} \geq \frac{1}{c} - \frac{1}{n}$ and

$$\sum_{\ell=\lfloor n/c \rfloor+1}^{n/d} \frac{1}{\ell - 1} = \sum_{\ell=\lfloor n/c \rfloor}^{n/d-1} \frac{1}{\ell} \geq \ln \frac{n/d}{\lfloor n/c \rfloor} \geq \ln \frac{c}{d}.$$

C.2.2. Proof of Lemma C.2

We prove Lemma C.2 in two steps. Observe that $\mathbb{E}[A_\ell \mid \neg E^{(\ell)}] = 0$ implies

$$\mathbb{E}[A_\ell] = \mathbb{E}\left[w(e^{(\ell)}) \mid E^{(\ell)}\right] \cdot Pr\left[E^{(\ell)}\right].$$

We proceed by showing, in Lemma C.3, that $\mathbb{E}\left[w(e^{(\ell)}) \mid E^{(\ell)}\right] \geq \frac{OPT}{n}$, and then in Lemma C.4 that $Pr\left[E^{(\ell)} \mid E^{(\ell)}\right] \geq \frac{\lfloor n/c \rfloor}{\ell - 1}$.

Let S be a subset of L of size ℓ, and let M_S be the optimum weighted matching w.r.t. the induced subgraph $G[S \cup N(S)]$. For a fixed subset $S \subseteq L$ with size ℓ, let $R_\ell(S)$ be the event that $\{$the node set of L_ℓ equals $S\}$, i.e. $\mathrm{Set}(L_\ell) = S$. Let $\mathcal{L}(\ell)$ be the set of all subsets of L of size ℓ, i.e., $\mathcal{L}(\ell) = \{S \subseteq L \ : \ |S| = \ell\}$.

Lemma C.3. *For every perfect bipartite graph $G = (L \cup R, E, w)$ it holds that*

$$\mathbb{E}\left[w(e^{(\ell)}) \mid E^{(\ell)}\right] \geq \frac{OPT}{n}.$$

Proof. Using conditional expectation,

$$\mathbb{E}\left[w(e^{(\ell)}) \mid E^{(\ell)}\right] = \sum_{S \in \mathcal{L}(\ell)} \mathbb{E}\left[w(e^{(\ell)}) \mid R_\ell(S) \wedge E^{(\ell)}\right] \cdot Pr\left[R_\ell(S)\right]. \tag{C.1}$$

Since the order of L is sampled u.a.r. we have $Pr\left[R_\ell(S)\right] = 1/\binom{n}{\ell}$, and thus it suffices to focus on the conditional expectation

$$\begin{aligned}\mathbb{E}\left[w(e^{(\ell)}) \mid R_\ell(S) \wedge E^{(\ell)}\right] &= \sum_{e^{(i)} \in M_S} w(e^{(i)}) \cdot Pr_{e^{(\ell)} \sim M_S}\left[e^{(\ell)} = e^{(i)}\right] \\ &= \frac{1}{\ell} \sum_{e^{(i)} \in M_S} w(e^{(i)}). \end{aligned} \tag{C.2}$$

where the last equality uses G is a perfect bipartite graph and Fact B.1. Then, by combining (C.1),(C.2) we have

$$\mathbb{E}\left[w(e^{(\ell)}) \mid E^{(\ell)}\right] = \frac{1}{\binom{n}{\ell}} \cdot \frac{1}{\ell} \sum_{S \in \mathcal{L}(\ell)} \sum_{e^{(i)} \in M_S} w(e^{(i)}). \tag{C.3}$$

Observe that for any subset $S \subseteq L$, it holds for $M^\star|_S = \{e^{(i)} = (i, r_i) \in M^\star \ : \ i \in S\}$ the restriction of the optimum matching $M^\star$ (w.r.t. the whole graph G) on S that

$$\sum_{e^{(i)} \in M_S} w(e^{(i)}) \geq \sum_{e^{(i)} \in M^\star|_S} w(e^{(i)}). \tag{C.4}$$

Further, since every vertex $i \in L(M^\star)$ appears in $\binom{n-1}{\ell-1}$ many subsets of size ℓ and $\binom{n-1}{\ell-1}/\binom{n}{\ell} = \ell/n$, it follows by (C.3),(C.4) that

$$\begin{aligned}\mathbb{E}\left[A_\ell \mid E^{(\ell)}\right] &\geq \frac{1}{\binom{n}{\ell}} \cdot \frac{1}{\ell} \sum_{S \in \mathcal{L}(\ell)} \sum_{e^{(i)} \in M^\star|_S} w(e^{(i)}) \\ &= \frac{\binom{n-1}{\ell-1}}{\binom{n}{\ell}} \cdot \frac{1}{\ell} \sum_{e^{(i)} \in M^\star} w(e^{(i)}) = \frac{OPT}{n}. \quad \square \end{aligned}$$

Lemma C.4. *For every perfect bipartite graph $G = (L \cup R, E, w)$ it holds that*

$$Pr\left[E^{(\ell)}\right] \geq \frac{\lfloor n/c \rfloor}{\ell - 1}. \tag{C.5}$$

Proof. For a fixed subset $S \subseteq L$ with size ℓ, let $F_{S,\ell}^{(i)}$ be the event that $\{$event $R_\ell(S)$ occurs$\}$ *and* $\{$the edge $e^{(\ell)} = (i, r_i)\}$. Then, by conditioning on the choice of subset $S \in \mathcal{L}(\ell)$, we have

$$\begin{aligned}Pr\left[E^{(\ell)}\right] &= \frac{1}{\binom{n}{\ell}} \sum_{S \in \mathcal{L}(\ell)} Pr\left[e^{(\ell)} \cup M \text{ is a matching} \mid R_\ell(S)\right] \\ &= \frac{1}{\binom{n}{\ell}} \frac{1}{\ell} \sum_{S \in \mathcal{L}(\ell)} \sum_{(i,r_i) \in M_S} Pr\left[(i, r_i) \cup M \text{ is a matching} \mid F_{S,\ell}^{(i)}\right]. \end{aligned} \tag{C.6}$$

Note that in (C.6), the subset $S \subseteq L$ with size ℓ and the edge $(i, r_i) \in M_S$ are fixed!

Given the first k nodes of L, in the order as they arrive, and the corresponding perfect matching (r.v.) $M^{(k)}$, let (r.v.) $e^{(k)} = (k, r_k)$ be the edge matched in (r.v.) $M^{(k)}$ from the last node (r.v.) k. Note that since G is perfect, it follows by Fact B.1 that edge $e^{(k)}$ exists and $|M^{(k)}| = k$. We denote by (r.v.) $M^{(k)}[k]$ the corresponding right node r_k.

Let Q_k be the event that

$$\{\text{node } r_i \notin M^{(k)}\} \vee \left\{\{\text{node } r_i \in M^{(k)}\} \wedge \{M^{(k)}[k] \neq r_i\}\right\}.$$

Then, we have

$$Pr\left[(i, r_i) \cup M \text{ is a matching} \mid F_{S,\ell}^{(i)}\right] = Pr\left[\bigwedge_{k=\lfloor n/c \rfloor+1}^{\ell-1} Q_k \mid F_{S,\ell}^{(i)}\right]. \tag{C.7}$$

Observe that the probability of event $\neg Q_k$ is equal to

$$\begin{aligned} & Pr\left[\{\text{node } r_i \in M^{(k)}\} \wedge \left\{\{\text{node } r_i \notin M^{(k)}\} \vee \{M^{(k)}[k] = r_i\}\right\}\right] \\ = & Pr\left[\{\text{node } r_i \in M^{(k)}\} \wedge \{M^{(k)}[k] = r_i\}\right]. \end{aligned} \tag{C.8}$$

Let $W_{i,t}$ be the event that $F^{(i)}_{S,\ell} \wedge \left(\bigwedge_{j=k}^{t-1} Q_j\right)$ for $t \in \{k+1, \dots, \ell-1\}$, and $W_{i,k}$ be the event $F^{(i)}_{S,\ell}$. Using conditional probability, we have

$$Pr\left[\bigwedge_{k=\lfloor n/c \rfloor+1}^{\ell-1} Q_k \mid F^{(i)}_{S,\ell}\right] = Pr\left[Q_{\ell-1} \mid W_{i,\ell-1}\right] \cdots Pr\left[Q_{k+1} \mid W_{i,k+1}\right] \cdot Pr\left[Q_k \mid W_{i,k}\right]. \tag{C.9}$$

We now analyze the terms in (C.9) separately. Let $\mathcal{T}(i,k)$ be the set of all matchings $M^{(k)}$ satisfying the event $F^{(i)}_{S,\ell} \wedge \{\text{node } r_i \in M^{(k)}\}$. Using (C.8), we have

$$\begin{aligned} Pr\left[\neg Q_k \mid F^{(i)}_{S,\ell}\right] &= Pr\left[\{\text{node } r_i \in M^{(k)}\} \wedge \{M^{(k)}[k] = r_i\} \mid F^{(i)}_{S,\ell}\right] \\ &\le Pr\left[M^{(k)}[k] = r_i \mid F^{(i)}_{S,\ell} \wedge \{\text{node } r_i \in M^{(k)}\}\right] \\ &= \frac{1}{|\mathcal{T}(i,k)|} \sum_{M' \in \mathcal{T}(i,k)} \frac{1}{|M'|} = \frac{1}{k}, \end{aligned} \tag{C.10}$$

where the last equality follows by Fact B.1 and G is perfect. Thus,

$$Pr\left[Q_k \mid W_{i,k}\right] = 1 - Pr\left[\neg Q_k \mid F^{(i)}_{S,\ell}\right] \ge 1 - \frac{1}{k}. \tag{C.11}$$

Similarly, for $t \in \{k+1, \dots, \ell-1\}$ we have

$$Pr\left[\neg Q_t \mid W_{i,t}\right] \le Pr\left[M^{(t)}[t] = r_i \mid W_{i,t} \wedge \{\text{node } r_i \in M^{(t)}\}\right] = \frac{1}{t},$$

and therefore

$$Pr\left[Q_t \mid W_{i,t}\right] = 1 - Pr\left[\neg Q_t \mid W_{i,t}\right] \ge 1 - \frac{1}{t}. \tag{C.12}$$

By combining (C.7),(C.9),(C.11),(C.12), we obtain

$$Pr\left[(i, r_i) \cup M \text{ is a matching} \mid F^{(i)}_{S,\ell}\right] \ge \prod_{k=\lfloor n/c \rfloor+1}^{\ell-1} \left(1 - \frac{1}{k}\right) = \frac{\lfloor n/c \rfloor}{\ell - 1}. \tag{C.13}$$

Since every summand in (C.6) is lower bounded by (C.13), we have

$$Pr\left[E^{(\ell)} \mid E^{(\ell)}_{\exists}\right] \ge \frac{\lfloor n/c \rfloor}{\ell - 1}. \quad \square$$

C.3. Algorithm 3 (Omitted Proofs)

We now lower bound the probability that an arbitrary node $r \in R$ remains unmatched after the completion of Phase II in Algorithm 3. Our analysis uses similar arguments as in Lemma C.4, but for the sake of completeness we present the proof below.

Lemma C.5. *For every constants $c \ge d \ge 1$ and for every perfect bipartite graph $G = (L \cup R, E, w)$, it holds for every node $r \in R$ that*

$$Pr\left[r \textit{ is not matched in } \text{Phase II}\right] \ge \frac{d}{c} - o(1).$$

Proof. Observe that

$$Pr\left[r \text{ is not matched in Phase II}\right] = Pr\left[\bigwedge_{k=\lfloor n/c\rfloor+1}^{\lfloor n/d\rfloor} Q_k\right].$$

Using (C.13), we have

$$Pr\left[r \text{ is not matched in Phase II}\right] \geq \prod_{k=\lfloor n/c\rfloor+1}^{\lfloor n/d\rfloor}\left(1-\frac{1}{k}\right) \geq \frac{\lfloor n/c\rfloor}{\lfloor n/d\rfloor} \geq \frac{d}{c}-\frac{d}{n}. \quad \square$$

Appendix D. Deterministic graphic matroid secretary algorithm

In this section, we analyze the competitive ratio of Algorithm 4.

Theorem 5.2. *The deterministic Algorithm 4 is* $(1/4 - o(1))$*-competitive for the graphic matroid secretary problem.*

The rest of this section is devoted to proving Theorem 5.2, and is organized as follows. In Appendix D.1, we give two useful summation closed forms. In Appendix D.2, we present our notation. In Appendix D.3, we extend Lemma C.2 to bipartite-matroid graphs. In Appendix D.4, we prove Theorem 5.2.

D.1. Summation bounds

Claim D.1. *For any* $k \in \mathbb{N}$ *and* $n \in \mathbb{N}_+$*, we have*

$$\sum_{\ell=n}^{n+k}\frac{1}{\ell\cdot(\ell+1)} = \frac{k+1}{n(n+k+1)}.$$

Proof. The proof is by induction. The base case follows by

$$\frac{1}{n}\cdot\frac{1}{n+1}+\frac{1}{n+1}\cdot\frac{1}{n+2} = \frac{2}{n(n+2)}.$$

Our inductive hypothesis is $\sum_{\ell=n}^{n+k}\frac{1}{\ell\cdot(\ell+1)} = \frac{k+1}{n(n+k+1)}$. Then, we have

$$\begin{aligned}\sum_{\ell=n}^{n+k+1}\frac{1}{\ell}\cdot\frac{1}{\ell+1} &= \frac{k+1}{n(n+k+1)}+\frac{1}{n+k+1}\cdot\frac{1}{n+k+2} = \frac{1}{n+k+1}\left[\frac{k+1}{n}+\frac{1}{n+k+2}\right]\\ &= \frac{1}{n+k+1}\left[\frac{(k+1)(n+k+1)+n+k+1}{n(n+k+2)}\right] = \frac{k+2}{n(n+k+2)}. \quad \square\end{aligned}$$

Claim D.2. *For any* $c > 1$*, it holds that*

$$f(c,n) := \frac{1}{n}\sum_{\ell=\lfloor n/c\rfloor}^{n-1}\frac{(\lfloor n/c\rfloor-1)\lfloor n/c\rfloor}{(\ell-1)\ell} = \frac{\lfloor n/c\rfloor}{n}\cdot\left[\frac{n-1}{n-2}-\frac{\lfloor n/c\rfloor}{n-2}\right] \gtrsim \frac{c-1}{c^2}.$$

In particular, the lower bound is maximized for $c = 2$ *and yields* $f(2,n) \gtrsim 1/4$.

Proof. By Claim D.1, we have

$$\sum_{\ell=\lfloor n/c\rfloor}^{n-1}\frac{1}{\ell-1}\cdot\frac{1}{\ell} = \sum_{\ell=\lfloor n/c\rfloor-1}^{\lfloor n/c\rfloor+[n-\lfloor n/c\rfloor-2]}\frac{1}{\ell}\cdot\frac{1}{\ell+1} = \frac{n-\lfloor n/c\rfloor-1}{(\lfloor n/c\rfloor-1)(n-2)},$$

and thus

$$\begin{aligned}\frac{1}{n}\sum_{\ell=\lfloor n/c\rfloor}^{n-1}\frac{(\lfloor n/c\rfloor-1)\lfloor n/c\rfloor}{(\ell-1)\ell} &= \frac{(\lfloor n/c\rfloor-1)\lfloor n/c\rfloor}{n}\cdot\frac{n-\lfloor n/c\rfloor-1}{(\lfloor n/c\rfloor-1)(n-2)}\\ &= \frac{\lfloor n/c\rfloor}{n}\cdot\left[\frac{n-1}{n-2}-\frac{\lfloor n/c\rfloor}{n-2}\right]\\ &\geq \left(\frac{1}{c}-\frac{1}{n}\right)\cdot\left[1+\frac{1}{n-2}-\frac{n}{n-2}\cdot\frac{1}{c}\right]\\ &\gtrsim \frac{1}{c}\cdot\left[1-\frac{1}{c}\right]=\frac{c-1}{c^2}.\end{aligned}$$

Let $g(x)=(x-1)/x^2$. Observe that its first derivative satisfies

$$\frac{d}{dx}g(x)=\frac{x^2-(x-1)2x}{x^4}=\frac{x(2-x)}{x^4}=0 \iff x_1=0 \quad x_2=2.$$

Further, $g(x)$ decreases in the range $[-\infty,0]$, increases in $[0,2]$ and again decreases in $[2,\infty]$. Hence, we have $\max_{x>0} g(x)=g(2)=1/4$. □

D.2. Notation

Given an undirected weighted graph $G'=(V,E',w')$, we construct the bipartite graph $B_G=(L\cup R,E)$ with weight function w as follows: Let the set of the right nodes be $R=V$; (ii) the set of the left nodes be $L=E'$, i.e., $\{u,v\}\in L$ if $\{u,v\}\in E'$; (iii) and for each edge in $\{u,v\}\in E'$ we insert two edges $\{\{u,v\},u\},\{\{u,v\},v\}\in E$ with equal weight $w(\{\{u,v\},u\})=w(\{\{u,v\},v\})=w'(\{u,v\})$. Note that although in Algorithm 4 $M^{(\ell)}$ is a (normal) matching, M is a special kind of matching, since for an edge $\{\{u,v\},u\}\in E$ (similarly $\{\{u,v\},v\}\in E$) to be matched it is required that both nodes $u,v\in R$ are not yet matched. To emphasize this, we refer to M as a *matching**.

D.3. Structural Lemma

We now extend Lemma C.2.

Lemma D.3. *Suppose $B_G=(L\cup R,E)$ is the bipartite graph representation of G with weight function w (as in Appendix D.2). Then, for every $c>1$ it holds for every $\ell\in\{\lfloor m/c\rfloor+1,\dots,m\}$ that*

$$\mathbb{E}[A_\ell]\geq\frac{\lfloor m/c\rfloor-1}{(\ell-1)-1}\cdot\frac{\lfloor m/c\rfloor}{\ell-1}\cdot\frac{OPT}{m}.$$

It is straightforward to verify that the analogue of Lemma C.3 holds, and yields

$$\mathbb{E}\left[w(e^{(\ell)})\mid E^{(\ell)}\right]\geq\frac{OPT}{m}.$$

Hence, to prove Lemma D.3 it remains to extend the statement of Lemma C.4.

Lemma D.4. *Suppose $B_G=(L\cup R,E)$ is the bipartite graph representation of G with weight function w (as in Appendix D.2). Then*

$$Pr\left[E^{(\ell)}\right]\geq\frac{\lfloor m/c\rfloor-1}{(\ell-1)-1}\cdot\frac{\lfloor m/c\rfloor}{\ell-1}.$$

Proof. We follow the proof in Lemma C.4, with the amendment that a node $\{u_k, v_k\} \in L$ and an edge $\{\{u_k, v_k\}, r_k\} \in E$. Recall that for a *fixed* subset $S \subseteq L$ with size ℓ and an edge $(i, r_i) \in M_S$, we can condition on the event $F_{S,\ell}^{(i)}$ that {the node set of L_ℓ equals S} *and* {the edge $e^{(\ell)} = (i, r_i)$}.

Let Q_k^r be the event that

$$\{\text{node } r \notin M^{(k)}\} \vee \left\{\{\text{node } r \in M^{(k)}\} \wedge \{M^{(k)}[k] \neq r\}\right\},$$

and let $\mathcal{P}_k$ be the event that $Q_k^{u_i} \wedge Q_k^{v_i}$. Then, we have

$$Pr\left[\{\{u_i, v_i\}, r_i\} \cup M \text{ is a matching}^\star \mid F_{S,\ell}^{(i)}\right] = Pr\left[\bigwedge_{k=\lfloor m/c\rfloor+1}^{\ell-1} \mathcal{P}_k \mid F_{S,\ell}^{(i)}\right].$$

Combining the Union bound and (C.10), yields

$$\begin{aligned} Pr\left[\neg\, \mathcal{P}_k \mid F_{S,\ell}^{(i)}\right] &= Pr\left[\neg\, Q_k^{u_i} \vee \neg\, Q_k^{v_i} \mid F_{S,\ell}^{(i)}\right] \\ &\leq Pr\left[\neg\, Q_k^{u_i} \mid F_{S,\ell}^{(i)}\right] + Pr\left[\neg\, Q_k^{v_i} \mid F_{S,\ell}^{(i)}\right] \\ &\leq \frac{2}{k}. \end{aligned}$$

Hence, using similar arguments as in the proof in Lemma C.4, we have

$$Pr\left[E^{(\ell)}\right] \geq \prod_{k=\lfloor m/c\rfloor+1}^{\ell-1} \left(1 - \frac{2}{k}\right) = \frac{\lfloor m/c\rfloor - 1}{(\ell-1)-1} \cdot \frac{\lfloor m/c\rfloor}{\ell-1}. \quad \square$$

D.4. Proof of Theorem 5.2

Using Lemma D.3, we have

$$\mathbb{E}\left[\sum_{\ell=1}^{m} A_\ell\right] = \sum_{\ell=\lfloor m/c\rfloor+1}^{m} \mathbb{E}\left[A_\ell\right] \geq \frac{OPT}{m} \sum_{\ell=\lfloor m/c\rfloor+1}^{m} \frac{\lfloor m/c\rfloor - 1}{(\ell-1)-1} \cdot \frac{\lfloor m/c\rfloor}{\ell-1}.$$

The statement follows by Claim D.2 and noting that

$$\frac{1}{m} \sum_{\ell=\lfloor m/c\rfloor+1}^{m} \frac{\lfloor m/c\rfloor - 1}{(\ell-1)-1} \cdot \frac{\lfloor m/c\rfloor}{\ell-1} \geq \left(\frac{c-1}{c^2} - o(1)\right) \geq \left(\frac{1}{4} - o(1)\right). \tag{D.1}$$

Appendix E. Graphic matroid secretary algorithm with predictions

In this section, we prove the worst-case bound of $(d-1)/c^2$ in Theorem 5.3, by analyzing Phase III of Algorithm 5.

Theorem E.1. *The Phase III in Algorithm 5 is* $(\frac{d-1}{c^2} - o(1))$*-competitive.*

The rest of this section is denoted to proving Theorem E.1, and is organized as follows. In Appendix E.1, we analyze the probability that a fixed pair of distinct vertices is eligible for matching in Phase III. In Appendix E.2, we give a lower bound on the event that $\{e^{(\ell)} \cup M$ is a matching$^\star\}$. In Appendix E.3, we prove Theorem E.1.

E.1. Pairwise node eligibility in Phase III

For any distinct nodes $u, v \in R$, we denote by $\Phi_{u,v}^{\notin M}$ the event that

$$\{u \text{ and } v \text{ are } \textit{not} \text{ matched in Phase II}\}.$$

Claim E.2. *It holds that*

$$Pr\left[\Phi_{u,v}^{\notin M}\right] \geq \left(\frac{d}{c}\right)^2 \cdot \frac{1-\frac{c}{m}}{1-\frac{d}{m}}.$$

Proof. Let S be a random variable denoting the set of all nodes in L that appear in Phase I and Phase II. Let

$$e'_{\max}(u, S) = \arg \max_{\{u,z\}\in S} w(u, z)$$

be a random variable denoting the node $\{u, z\} \in L$ with largest weight seen in the set S.

The proof proceeds by case distinction:

Case 1. Suppose $e'_{\max}(u, S) = e'_{\max}(v, S)$, i.e., there is a node $\{u, v\} \in S$. Let $\mathcal{K}_r(S)$ be the event that node $e'_{\max}(r, S) \in S$ is sampled in Phase I. By conditioning on the choice of S, we have

$$Pr\left[\Phi_{u,v}^{\notin M}\right] = \sum_S Pr\left[\mathcal{K}_r(S) \mid S\right] \cdot Pr\left[S\right] = \frac{m/c}{m/d} \cdot \sum_S Pr\left[S\right] = \frac{d}{c}.$$

Case 2. Suppose $e'_{\max}(u, S) \neq e'_{\max}(v, S)$, i.e., there are distinct nodes $\{u, x\} \in S$ and $\{v, y\} \in S$ with largest weight, respectively from u and v. By conditioning on the choice of S, we have

$$Pr\left[\mathcal{K}_u(S) \wedge \mathcal{K}_v(S) \mid S\right] = \frac{2\binom{\frac{m}{c}}{2} \cdot (\frac{m}{d}-2)!}{(\frac{m}{d})!} = \left(\frac{d}{c}\right)^2 \cdot \frac{1-\frac{c}{m}}{1-\frac{d}{m}},$$

and thus

$$Pr\left[\Phi_{u,v}^{\notin M}\right] = \sum_S Pr\left[\mathcal{K}_u(S) \wedge \mathcal{K}_v(S) \mid S\right] \cdot Pr\left[S\right] = \left(\frac{d}{c}\right)^2 \cdot \frac{1-\frac{c}{m}}{1-\frac{d}{m}}. \quad \square$$

E.2. Lower bounding the Matching* event

Recall that $E^{(\ell)}$ denotes the event that $\{e^{(\ell)} \cup M$ is a matching$^\star\}$, where (r.v.) M is the current online matching*, see Appendix D.2 for details. In order to control the possible negative side effect of selecting suboptimal edges in Phase II, we extend Lemma D.4 and give a lower bound on the event $E^{(\ell)}$ for any node $\ell \in L$ that appears in Phase III.

Lemma E.3. *Suppose $B_G = (L \cup R, E)$ is the bipartite graph representation of G with weight function w (as in Appendix D.2). Algorithm 5 guarantees in Phase III that*

$$Pr\left[E^{(\ell)}\right] \geq \frac{\lfloor m/d \rfloor - 1}{(\ell-1)-1} \cdot \frac{\lfloor m/d \rfloor}{\ell - 1} \cdot Pr\left[\Phi_{u,v}^{\notin M}\right], \quad \forall \ell \in \{\lfloor m/d \rfloor + 1, \ldots, m\}.$$

Proof. We follow the proof in Lemma C.4, with the amendment that a node $\{u_k, v_k\} \in L$ and an edge $\{\{u_k, v_k\}, r_k\} \in E$. Recall that for a *fixed* subset $S \subseteq L$ with size ℓ and a *fixed* edge $\{\{u_i, v_i\}, r_i\} \in M_S$, we can condition on the event $F_{S,\ell}^{(i)}$ that

$$\{\text{the set of nodes of } L_\ell \text{ equals } S\} \text{ and } \{\text{the edge } e^{(\ell)} = \{\{u_i, v_i\}, r_i\}\}.$$

Let Q_k^r be the event that

$$\{\text{node } r \notin M^{(k)}\} \vee \left\{\{\text{node } r \in M^{(k)}\} \wedge \{M^{(k)}[k] \neq r\}\right\},$$

and let $\mathcal{P}_k$ denotes the event $Q_k^{u_i} \wedge Q_k^{v_i}$. The proof proceeds by case distinction:

Case 1. For $\ell = \lfloor m/d \rfloor + 1$, we have

$$Pr\left[E^{(\ell)}\right] = Pr\left[\{\{u_i, v_i\}, r_i\} \cup M \text{ is a matching}^\star \mid F_{S,\ell}^{(i)}\right] = Pr\left[\Phi_{u,v}^{\notin M}\right].$$

Case 2. For $\ell = \{\lfloor m/d \rfloor + 2, \ldots, m\}$, we have

$$\begin{aligned} Pr\left[E^{(\ell)}\right] &= Pr\left[\{\{u_i, v_i\}, r_i\} \cup M \text{ is a matching}^\star \mid F_{S,\ell}^{(i)}\right] \\ &= Pr\left[\bigwedge_{k=\lfloor m/d \rfloor+1}^{\ell-1} \mathcal{P}_k \mid F_{S,\ell}^{(i)} \wedge \Phi_{u,v}^{\notin M}\right] \cdot Pr\left[\Phi_{u,v}^{\notin M}\right]. \end{aligned}$$

Combining the Union bound and (C.10), yields

$$\begin{aligned} Pr\left[\neg \mathcal{P}_k \mid F_{S,\ell}^{(i)} \wedge \Phi_{u,v}^{\notin M}\right] &= Pr\left[\neg Q_k^{u_i} \vee \neg Q_k^{v_i} \mid F_{S,\ell}^{(i)} \wedge \Phi_{u,v}^{\notin M}\right] \\ &\leq Pr\left[\neg Q_k^{u_i} \mid F_{S,\ell}^{(i)} \wedge \Phi_{u,v}^{\notin M}\right] + Pr\left[\neg Q_k^{v_i} \mid F_{S,\ell}^{(i)} \wedge \Phi_{u,v}^{\notin M}\right] \\ &\leq \frac{2}{k}. \end{aligned}$$

Hence, using similar arguments as in the proof of Lemma C.4, we have

$$Pr\left[\bigwedge_{k=\lfloor m/d \rfloor+1}^{\ell-1} \mathcal{P}_k \mid F_{S,\ell}^{(i)} \wedge \Phi_{u,v}^{\notin M}\right] \geq \prod_{k=\lfloor m/d \rfloor+1}^{\ell-1} \left(1 - \frac{2}{k}\right) = \frac{\lfloor m/d \rfloor - 1}{(\ell-1)-1} \cdot \frac{\lfloor m/d \rfloor}{\ell - 1}.$$

Therefore, it holds that

$$Pr\left[E^{(\ell)}\right] \geq \frac{\lfloor m/d \rfloor - 1}{(\ell-1)-1} \cdot \frac{\lfloor m/d \rfloor}{\ell - 1} \cdot Pr\left[\Phi_{u,v}^{\notin M}\right]. \quad \square$$

E.3. Proof of Theorem 5.3

In this section, we analyze the expected contribution in Phase III. Our goal now is to lower bound the expression

$$\mathbb{E}\left[A_\ell\right] = \mathbb{E}\left[w(e^{(\ell)}) \mid E^{(\ell)}\right] \cdot Pr\left[E^{(\ell)}\right].$$

It is straightforward to verify that Lemma C.3 holds in the current setting and yields

$$\mathbb{E}\left[w(e^{(\ell)}) \mid E^{(\ell)}\right] \geq \frac{OPT}{m}.$$

Further, by Lemma E.3, it holds for every $\ell \in \{\lfloor m/d \rfloor + 1, \ldots, m\}$ that

$$\begin{aligned} \mathbb{E}\left[A_\ell\right] &= \mathbb{E}\left[w(e^{(\ell)}) \mid E^{(\ell)}\right] \cdot Pr\left[E^{(\ell)}\right] \\ &\geq \frac{OPT}{m} \cdot \frac{\lfloor m/d \rfloor - 1}{(\ell-1)-1} \cdot \frac{\lfloor m/d \rfloor}{\ell - 1} \cdot Pr\left[\Phi_{u,v}^{\notin M}\right], \end{aligned}$$

and thus

$$\sum_{\ell=\lfloor m/d \rfloor+1}^{m} \mathbb{E}\left[A_\ell\right] \geq Pr\left[\Phi_{u,v}^{\notin M}\right] \cdot \frac{OPT}{m} \sum_{\ell=\lfloor m/d \rfloor+1}^{m} \frac{\lfloor m/d \rfloor - 1}{(\ell-1)-1} \cdot \frac{\lfloor m/d \rfloor}{\ell - 1}.$$

Hence, by combining the first inequality in (D.1) and Claim E.2, yields

$$\mathbb{E}\left[\sum_{\ell=\lfloor m/d\rfloor+1}^{m} A_\ell\right] \geq \left(\frac{d}{c}\right)^2 (1-o(1)) \cdot \left(\frac{d-1}{d^2} - o(1)\right) \cdot OPT$$
$$\geq \left(\frac{d-1}{c^2} - o(1)\right) \cdot OPT.$$

References

[1] T. Kraska, A. Beutel, E.H. Chi, J. Dean, N. Polyzotis, The case for learned index structures, in: Proc. of the 2018 International Conference on Management of Data, SIGMOD, 2018, pp. 489–504.
[2] E.B. Khalil, B. Dilkina, G.L. Nemhauser, S. Ahmed, Y. Shao, Learning to run heuristics in tree search, in: Proc. of the 26th International Joint Conference on Artificial Intelligence, in: (IJCAI), 2017, pp. 659–666.
[3] D. Rohatgi, Near-optimal bounds for online caching with machine learned advice, in: Proc. of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA, 2020, pp. 1834–1845.
[4] S. Lattanzi, T. Lavastida, B. Moseley, S. Vassilvitskii, Online scheduling via learned weights, in: Proc. of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA, 2020, pp. 1859–1877.
[5] T. Lykouris, S. Vassilvitskii, Competitive caching with machine learned advice, in: Proc. of the 35th International Conference on Machine Learning, (ICML), Vol. 80, 2018, pp. 3302–3311.
[6] M. Purohit, Z. Svitkina, R. Kumar, Improving online algorithms via ML predictions, in: Proc. of Advances in Neural Information Processing Systems, (NeurIPS), 2018, pp. 9661–9670.
[7] S. Gollapudi, D. Panigrahi, Online algorithms for rent-or-buy with expert advice, in: Proc. of 36th International Conference of Machine Learning, (ICML), 2019, pp. 2319–2327.
[8] M. Mitzenmacher, Scheduling with predictions and the price of misprediction, in: Proc. of the 11th Innovations in Theoretical Computer Science Conference, (ITCS), in: LIPIcs, vol. 151, 2020, pp. 14:1–14:18.
[9] A.M. Medina, S. Vassilvitskii, Revenue optimization with approximate bid predictions, in: Proc. of the 31st International Conference on Neural Information Processing Systems, (NeurIPS), 2017, pp. 1856–1864.
[10] M. Mitzenmacher, S. Vassilvitskii, Algorithms with predictions, Commun. ACM 65 (7) (2022) 33–35.
[11] A. Lindermayr, N. Megow (creators), Algorithms with predictions. URL https://algorithms-with-predictions.github.io/.
[12] M. Gardner, Mathematical games, Sci. Am. (1960) 150–153.
[13] D.V. Lindley, Dynamic programming and decision theory, J. R. Stat. Soc. 10 (1) (1961) 39–51.
[14] E.B. Dynkin, The optimum choice of the instant for stopping a Markov process, Sov. Math. Dokl 4 (1962).
[15] N. Korula, M. Pál, Algorithms for secretary problems on graphs and hypergraphs, in: Proc. of 36th International Colloquium on Automata, Languages, and Programming, (ICALP), 2009, pp. 508–520.
[16] T. Kesselheim, K. Radke, A. Tönnis, B. Vöcking, An optimal online algorithm for weighted bipartite matching and extensions to combinatorial auctions, in: Proc. of 21st European Symposium on Algorithms, (ESA), 2013, pp. 589–600.
[17] G. Aggarwal, G. Goel, C. Karande, A. Mehta, Online vertex-weighted bipartite matching and single-bid budgeted allocations, in: Proc. of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms, (SODA), 2011, pp. 1253–1264.
[18] J.A. Soto, A. Turkieltaub, V. Verdugo, Strong algorithms for the ordinal matroid secretary problem, in: Proc. of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms, 2018, pp. 715–734.
[19] M. Babaioff, N. Immorlica, D. Kempe, R. Kleinberg, Matroid secretary problems, J. ACM 65 (6) (2018).
[20] D. Bradac, A. Gupta, S. Singla, G. Zuzic, Robust algorithms for the secretary problem, in: Proc. of 11th Conference on Innovations in Theoretical Computer Science Conference (ITCS), Vol. 151, 2020, pp. 32:1–32:26.
[21] A. Fiat, Y. Rabani, Y. Ravid, Competitive k-server algorithms, J. Comput. System Sci. 48 (3) (1994) 410–428.
[22] M. Mahdian, H. Nazerzadeh, A. Saberi, Online optimization with uncertain information, ACM Trans. Algorithms 8 (1) (2012) 2:1–2:29.
[23] O. Lachish, O (log log rank) competitive ratio for the matroid secretary problem, in: Proc. of 55th Annual Symposium on Foundations of Computer Science, (FOCS), 2014, pp. 326–335.
[24] M. Feldman, O. Svensson, R. Zenklusen, A simple o (log log (rank))-competitive algorithm for the matroid secretary problem, in: Proc. of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms, (SODA), 2014, pp. 1189–1201.
[25] N.B. Dimitrov, C.G. Plaxton, Competitive weighted matching in transversal matroids, Algorithmica 62 (1) (2012) 333–348.
[26] M. Babaioff, M. Dinitz, A. Gupta, N. Immorlica, K. Talwar, Secretary problems: weights and discounts, in: Proc.of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms, 2009, pp. 1245–1254.
[27] T. Kesselheim, R. Kleinberg, R. Niazadeh, Secretary problems with non-uniform arrival order, in: Proc. of the 47th Annual ACM Symposium on Theory of Computing, (STOC), 2015, pp. 879–888.
[28] P. Garg, S. Kale, L. Rohwedder, O. Svensson, Robust algorithms under adversarial injections, 2020, arXiv preprint arXiv:2004.12667.
[29] H. Kaplan, D. Naori, D. Raz, Competitive analysis with a sample and the secretary problem, in: Proc. of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms, (SODA), 2020, pp. 2082–2095.
[30] J. Correa, A. Cristi, B. Epstein, J. Soto, Sample-driven optimal stopping: From the secretary problem to the iid prophet inequality, 2020, arXiv preprint arXiv:2011.06516.

[31] J. Correa, A. Cristi, L. Feuilloley, T. Oosterwijk, A. Tsigonias-Dimitriadis, The secretary problem with independent sampling, in: Proc. of the 32nd ACM-SIAM Symposium on Discrete Algorithms, SODA, 2021, pp. 2047–2058.
[32] B. Lucier, An economic view of prophet inequalities, ACM SIGecom Exch. 16 (1) (2017) 24–47.
[33] J. Correa, P. Foncea, R. Hoeksma, T. Oosterwijk, T. Vredeveld, Recent developments in prophet inequalities, ACM SIGecom Exch. 17 (1) (2019) 61–70.
[34] U. Krengel, L. Sucheston, On semiamarts, amarts, and processes with finite value, Adv. Appl. Probab. 4 (197–266) (1978) 1–5.
[35] R. Kleinberg, S.M. Weinberg, Matroid prophet inequalities and applications to multi-dimensional mechanism design, Games Econom. Behav. 113 (2019) 97–115.
[36] S. Ehsani, M. Hajiaghayi, T. Kesselheim, S. Singla, Prophet secretary for combinatorial auctions and matroids, in: Proc. of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms, (SODA), 2018, pp. 700–714.
[37] P.D. Azar, R. Kleinberg, S.M. Weinberg, Prophet inequalities with limited information, in: Proc. of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms, (SODA), 2014, pp. 1358–1377.
[38] J. Wang, The prophet inequality can be solved optimally with a single set of samples. arXiv Preprint arXiv:1812.10563.
[39] J. Correa, P. Dütting, F. Fischer, K. Schewior, Prophet inequalities for I.I.D. Random variables from an unknown distribution, in: Proc. of the 20th ACM Conference on Economics and Computation, (EC), 2019, pp. 3–17.
[40] J.R. Correa, A. Cristi, B. Epstein, J.A. Soto, The two-sided game of googol and sample-based prophet inequalities, in: Proc. of the 31st ACM-SIAM Symposium on Discrete Algorithms, (SODA), 2020, pp. 2066–2081.
[41] P. Dütting, T. Kesselheim, Posted pricing and prophet inequalities with inaccurate priors, in: Proc. of the 20th ACM Conference on Economics and Computation, (EC), 2019, pp. 111–129.
[42] P. Dütting, S. Lattanzi, R. Paes Leme, S. Vassilvitskii, Secretaries with advice, in: Proc. of the 22nd ACM Conference on Economics and Computation, EC, 2021, pp. 409–429.
[43] R.M. Karp, U.V. Vazirani, V.V. Vazirani, An optimal algorithm for on-line bipartite matching, in: Proc. of the 22nd Annual ACM Symposium on Theory of Computing, (STOC), 1990, pp. 352–358.
[44] M. Mahdian, Q. Yan, Online bipartite matching with random arrivals: an approach based on strongly factor-revealing lps, in: Proc. of the 43rd Annual ACM Symposium on Theory of Computing, (STOC), 2011, pp. 597–606.
[45] Z. Huang, Z.G. Tang, X. Wu, Y. Zhang, Online vertex-weighted bipartite matching: Beating 1-1/e with random arrivals, ACM Trans. Algorithms (TALG) 15 (3) (2019) 1–15.