# Price-setting Problems and Matroid Bayesian Online Selection

by

Ian DeHaan

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Combinatorics and Optimization

Waterloo, Ontario, Canada, 2024

© Ian DeHaan 2024

## Author's Declaration

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

## Statement of Contributions

All research detailed in this thesis was completed in collaboration with my supervisor, Kanstantsin Pashkovich. Most of Chapter 1, Chapter 2, and Section 3.1 are adapted from our co-authored paper [DP24].

## Abstract

We study a class of Bayesian online selection problems with matroid constraints. Consider a seller who has several items they wish to sell, with the set of sold items being subject to some structural constraints, e.g., the set of sold items should be independent with respect to some matroid. Each item has an offer value drawn independently from a known distribution. In a known order, items arrive and drawn values are presented to the seller. If the seller chooses to sell the item, they gain the drawn value as revenue. Given distribution information for each item, the seller wishes to maximize their expected revenue by carefully choosing which offers to accept as they arrive.

Such problems have been studied extensively when the seller's revenue is compared with the offline optimum, referred to as the "prophet". In this setting, a tight 2-competitive algorithm is known when the seller is limited to selling independent sets of a matroid [KW12]. We turn our attention to the online optimum, or "philosopher", and ask how well the seller can do with polynomial-time computation, compared to a seller with unlimited computation but with the same limited distribution information about offers.

We show that when the underlying constraints are laminar and the arrival of buyers follows a natural "left-to-right" order, there is a polynomial-time approximation scheme for maximizing the seller's revenue. We also show that such a result is impossible for the related case when the underlying constraints correspond to a graphic matroid. In particular, it is PSPACE-hard to approximate the philosopher's expected revenue to some fixed constant $\alpha < 1$; moreover, this cannot be alleviated by requirements on the arrival order in the case of graphic matroids. We also show similar hardness results for both transversal and cographic matroids.

We then turn our attention to a related problem where the arrival order of items is unknown and uniformly random. In this setting, we show that there is a polynomial-time approximation scheme whenever the rank of the matroid is bounded above by a constant and all probabilities in the input are bounded below by a constant.

We additionally examine the computational complexity of computing the prophet's expected revenue. We show that this problem is #P-hard, and give a fully polynomial-time randomized approximation scheme for the problem.

**Acknowledgments**

To Kanstantsin Pashkovich, thank you for being the best supervisor I could have asked for. Your passion for and the joy you find in mathematics is contagious. I have learned so much from you over the last two years. Thank you for your kindness during difficult times, for your consistently helpful advice, and for your flexibility around my external hobbies. Thank you, Kostya, for helping me grow into a researcher.

To Zac Friggstad back at the University of Alberta, thank you for introducing me to theoretical computer science and competitive programming all those years ago. Some of my fondest memories are of discussing problems with you while traveling to contests. Thank you for all the helpful advice you've given me, especially over the last two years. My path to research was greatly influenced by you and I'm grateful to have had the opportunity to work with you.

To Sepehr Assadi and Chaitanya Swamy, thank you for taking the time to read and comment on my thesis. Your suggestions were incredibly helpful and have increased the quality of this thesis.

To Nat, thank you for all the time we've spent together over the last 5 years. You have made my life immeasurably better. I wish you the best of luck on your future endeavors in Germany. You have been and will remain a very important part of my life.

To my friends in the Waterloo region, thank you for being there for me throughout these last couple of years. I'll never forget the trivia nights, strange food creations in the campus plaza, or late-night study sessions. Special thanks go to Jacob, Joseph, Nat, Noah G, and Noah W. Our game nights still live in my heart and I hope there will be more games of innovation to come.

To the members of Quote, especially those of you who I had the opportunity to meet in-person over the last year, thank you. You have all been a steady source of encouragement and entertainment for the last seven years.

# Table of Contents

**Author's Declaration** ii

**Statement of Contributions** iii

**Abstract** iv

**Acknowledgments** v

**1 Introduction** 1

1.1 Our Results . . . 4

1.2 Overview of Techniques . . . 7

1.2.1 PTAS for Left-to-Right Laminar Bayesian Online Selection . . . 7

1.2.2 Hardness of Matroid Bayesian Online Selection . . . 9

1.2.3 PTAS for Constant-Rank Prophet Secretary . . . 11

1.3 Overview of Related Work . . . 11

1.4 Preliminaries . . . 12

**2 PTAS for Left-to-Right Laminar Bayesian Online Selection** 14

2.1 Preliminaries . . . 15

2.2 Warm-Up: QPTAS . . . 15

2.3 Linear Programming Formulation and Rounding Algorithm . . . 18

2.3.1 Small and Big Bins . . . 18

2.3.2 Feasibility on Small Bins . . . 19
2.3.3 Feasibility on Big Bins . . . 20
2.3.4 Algorithm . . . 21
2.4 Analysis . . . 21
2.4.1 Losses from Discarding Elements in Step 6 of Algorithm 1 . . . 22
2.4.2 Losses from Transformations in Steps 1-3 of Algorithm 1 . . . 23
2.4.3 Characterizations of Policies in Step 5 of Algorithm 1 . . . 24
2.4.4 Properties of Optimal Online Policies on Left-to-Right LBOS . . . 25
2.4.5 Results to Prove Concentration . . . 28
2.4.6 Putting Everything Together . . . 30
2.5 Extending to Arrivals "Close" to Left-to-Right . . . 30
2.6 Laminar Instances with Bad Concentration . . . 31

**3 Hardness of Matroid Bayesian Online Selection 37**
3.1 Graphic Matroids . . . 38
3.1.1 Hardness for Graphic Bayesian Online Selection . . . 38
3.1.2 No PTAS-Compatible Distribution-Agnostic Order . . . 47
3.2 Transversal Matroids . . . 47
3.2.1 Reduction from MAX-S2SAT to TBOS . . . 48
3.3 Co-Graphic Matroids . . . 49

**4 Prophet Secretary 54**
4.1 PTAS for High-Probability Constant-Rank Matroids . . . 55
4.1.1 Preprocessing . . . 55
4.1.2 Warm-up: The Single-Item Case . . . 57
4.1.3 PTAS for the More General Case . . . 59

**5 Computing the Prophet's Revenue** **63**

5.1 Hardness in Planar Graphic Matroids . . . . . 63

5.2 FPRAS for all Matroids . . . . . 65

5.3 Exact Algorithm for Laminar Matroids . . . . . 67

**References** **70**

# Chapter 1

# Introduction

Suppose that someone is trying to sell a house. They have arranged viewings with $n$ buyers, the $i$th of which will view the house $i$ days from today. At the end of the $i$th viewing, the $i$th buyer will offer some amount of money for the house. The seller must immediately choose to either accept the offer or irrevocably turn the offer down.

The seller does not have perfect information on how much money each buyer will offer. The seller knows only a distribution $F_i$ for the $i$th buyer's offer. After the $i$th viewing, the seller is offered an independently sampled value $v_i \sim F_i$ by the $i$th buyer. How should the seller act to maximize their expected revenue? Should the seller accept an offer of $v_i$ from the $i$th buyer? This is equivalent to finding the minimum value of $v_i$ at which the seller should accept the $i$th buyer's offer, i.e., the price the seller should set for buyer $i$.

Study of this problem, known as the single-item prophet inequality problem, was initiated by Krengel and Sucheston [KS77]. They compared how well the seller can perform in relation to the eponymous *prophet*, a seller who knows all offers ahead of time and simply accepts the highest offer. Later, Samuel-Cahn showed that the seller can do at least half as well as the prophet with a simple threshold strategy [SC84]. This can be achieved by accepting the first offer with value at least half the expected maximum offer $\mathbb{E}[\max_i v_i]/2$.

This simple strategy of [SC84] is optimal due to the instance in Figure 1.1. Regardless of the seller's choice on the first buyer's offer, they will obtain a revenue of at most 1 in expectation. Meanwhile, the prophet can choose whether to accept the first buyer's offer after observing the second buyer's offer. Therefore, the prophet will earn a revenue of $\frac{1}{\epsilon}$ whenever the second buyer offers it, and a revenue of 1 from the first buyer otherwise. This yields an expected revenue of $\epsilon \cdot \frac{1}{\epsilon} + (1-\epsilon) \cdot 1 = 2 - \epsilon$ for the prophet. Letting $\epsilon$ tend to 0, we see that the seller cannot guarantee better than half of the prophet's expected revenue.

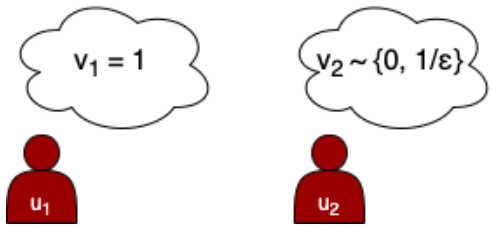



Figure 1.1: The first buyer offers 1 always. The second buyer offers $\frac{1}{\epsilon}$ with probability $\epsilon$, and 0 otherwise.

In contrast to the standard approximation algorithms literature, the hardness here is due to lack of *knowledge*, not lack of *computational power*. In fact, the "optimal" strategy for the seller is very simple in this scenario. The seller can simply always accept the first buyer's offer to maximize their expected revenue. Despite this simplicity, the seller cannot even get close to matching the benchmark of the prophet. Indeed, despite the mechanism from [SC84] achieving the best possible performance ratio in comparison to the prophet, it can perform badly in comparison to the *optimal online strategy*. In Figure 1.2, it is clear that both the prophet and the optimal online strategy will achieve a revenue of 2. But the mechanism from [SC84] will accept the first offer and obtain a revenue of only 1.

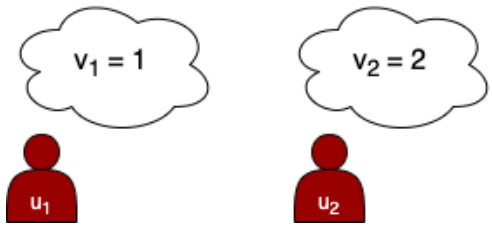



Figure 1.2: Buyer 1 and 2 deterministically offer values 1 and 2, respectively.

To address these shortcomings, we turn to the benchmark of the *philosopher*, or online optimum. This philosopher has unlimited computational power but is limited to the same knowledge that the seller has. The question now becomes how well can a seller limited to polynomial-time computation perform in relation to the philosopher? This benchmark has historically been referred to as the online optimum, while the philosopher term was introduced recently in [BDP+24]. For simplicity, suppose that all probability distributions in the input are represented as a list of probability, value pairs. In the single-item case, there is a simple dynamic programming algorithm that allows the philosopher's expected revenue to be computed exactly in polynomial-time. This allows a seller limited to polynomial-time

computation to match the philosopher's strategy and expected revenue.

While this and the result from [SC84] essentially close the book on study of the single-item prophet-inequality problem, many generalizations and related problems have been considered in the decades since. In particular, there has been much progress made on instances where the seller is offering multiple items for sale, with constraints on which subsets of items are allowed to be sold.

Prominent among these generalizations is the *matroid prophet inequality* problem, in which the seller is offering items from a matroid, and is limited to selling independent sets. In this problem, each buyer is interested in one specific item of the matroid. Surprisingly, the seller can do just as well in comparison to the prophet in the matroid prophet inequality problem as in the single-item case discussed above. There exists a mechanism by which the seller can guarantee themselves at least half the expected revenue of the prophet in expectation [KW12]. The naive implementation of the mechanism from [KW12] results in an exponential-time algorithm, which raises the natural question of what can be done in polynomial-time. To focus completely on hardness due to computation, we use the benchmark of the philosopher. In this thesis, we address this question in numerous special cases, including laminar, graphic, cographic, and transversal matroids. As the prophet inequality name starts to lose meaning when the prophet benchmark is not in use, we refer to approximating the online optimum in matroid prophet inequalities as *matroid Bayesian online selection*.

It may seem strange that the philosopher benchmark has only been considered widely in recent years. One explanation for this is the technical difficulty of understanding even basic properties of the philosopher benchmark, which stands in stark contrast to the wide-reaching results known about the prophet benchmark. As stated in Chapter 24 of [Rou21] and quoted by [CST24],

> "Although using the optimal online algorithm as a benchmark makes perfect sense in stochastic settings, it is not so common in the literature, largely because we do not have many techniques for getting a handle on the optimal online algorithm or comparing arbitrary online algorithms in stochastic settings."

Indeed, as we will see in Section 2.6, the optimal online algorithm for matroid Bayesian online selection often exhibits strange behavior that is difficult to design algorithms around.

We are not the first to consider this problem — in 2017, Gupta posed the question of whether the philosopher's expected revenue can be computed in polynomial-time for all matroids, and more specifically for the special cases of graphic and laminar matroids

[Gup17]. In 2019, Anari et al. [ANSS19] considered two special cases of *laminar Bayesian online selection*, the special case of matroid Bayesian online selection in which the underlying matroid is laminar. They gave a polynomial-time approximation scheme for each case.

## 1.1 Our Results

In this thesis, we answer the question of Gupta [Gup17] in the negative, showing in Chapter 3 that it is PSPACE-hard to approximate the philosopher's expected revenue to some fixed constant $\alpha < 1$ in graphic matroids.

In Chapter 2, we give a polynomial-time approximation scheme (PTAS) for the philosopher's expected revenue in all laminar matroids with so-called "left-to-right" arrival orders. Intuitively, left-to-right arrival orders are those in which the elements of each "constraint" arrive consecutively. Furthermore, the provided algorithm also holds for arrival orders that are "close" to left-to-right orders.

Let us provide an example of a Bayesian selection problem with a left-to-right arrival order. Consider a situation where items correspond to "clients", and the seller is interested in determining which clients to serve. The seller knows clients' arrival order and distributions for their offers. Let the parameter $p$ represent some crucial resource and so determine restrictions on the number of clients the seller can serve. In particular, let there be several critical thresholds $p_1$, $p_2$, ... for $p$ and limits $\gamma_1$, $\gamma_2$, .... For each $i$, if the value of $p$ drops below $p_i$ at some timestamp, then the seller can serve at most $\gamma_i$ clients between this timestamp and the next timestamp when the value of $p$ is again at least $p_i$. In Figure 1.3, one can see the example of how the parameter $p$ changes over time and the corresponding laminar family. The arrival order in this problem is left-to-right due to the items arriving in order of time. Since each constraint corresponds to an interval of time, the items contained in that constraint must arrive consecutively. We note that the arrival order in the "production constrained Bayesian selection" from [ANSS19] corresponds to the case when $p$ is a resource that is being delivered to us over time, see Figure 1.4, plus one additional global constraint.

Our algorithm in Chapter 2 works for more general matroid Bayesian online selection problems than the ones described above. We can handle the cases where the limits $\gamma_1$, $\gamma_2$, ... are functions of timestamps when the value of $p$ drops below $p_i$. Moreover, we allow the arrival order to be more complicated than just left-to-right. For example, one can imagine that the seller runs several services in parallel, which mostly depend on different resources,

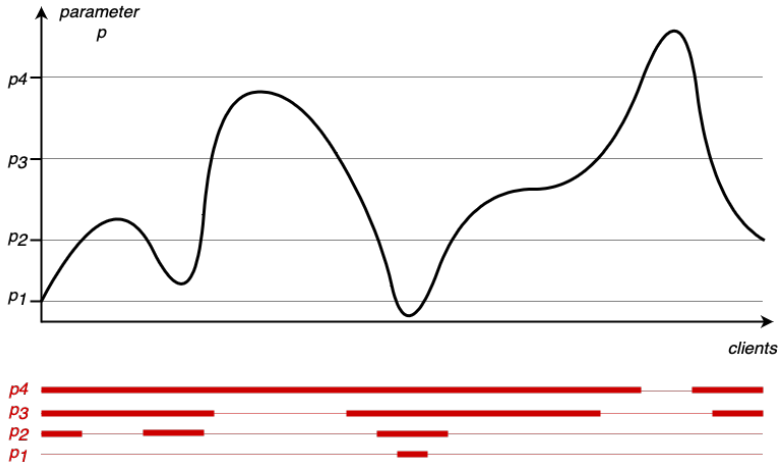



Figure 1.3: Here, the horizontal axis is associated with clients and so with the timestamps, since their arrivals provide a measure of time. The vertical axis is associated with the value of the parameter $p$. The critical thresholds $p_1$, $p_2$, $p_3$, and $p_4$ for the value of $p$ are depicted on the vertical axis. Below the picture of the graph, one can find the illustration for the corresponding laminar matroid. In particular, the picture below contains an interval for each critical threshold and the timestamps, when the value of $p$ drops below the threshold and the next timestamp when the value of $p$ reaches the value of the threshold.

however each type of service depends on at most a constant number of resources that are needed also for other types of service. In this situation, as long as the dependence on the "overarching" resources leads to a laminar matroid, our algorithm provides a PTAS for the arising Bayesian online selection problem.

Thus, our results provide a generalization and extension of the previous results for laminar matroids. The study of laminar Bayesian online selection problems was initiated by Anari et al. [ANSS19]. They gave a PTAS for the special cases of bounded-depth laminar matroids and production constrained selection. Note that both of these special cases have required structure on the laminar family, with production constrained selection also having a special arrival order. In contrast, our results do not impose any requirements on the structure of the laminar matroids, but only on the arrival order.

Unfortunately, our techniques do not allow us to say anything about the laminar Bayesian online selection when there are no restrictions on the arrival order. Our results require some notion of "concentration" on the number of elements selected by the philosopher. We show in Section 2.6 that there exist laminar Bayesian online selection instances with essentially worst-possible concentration. Interestingly, these instances are

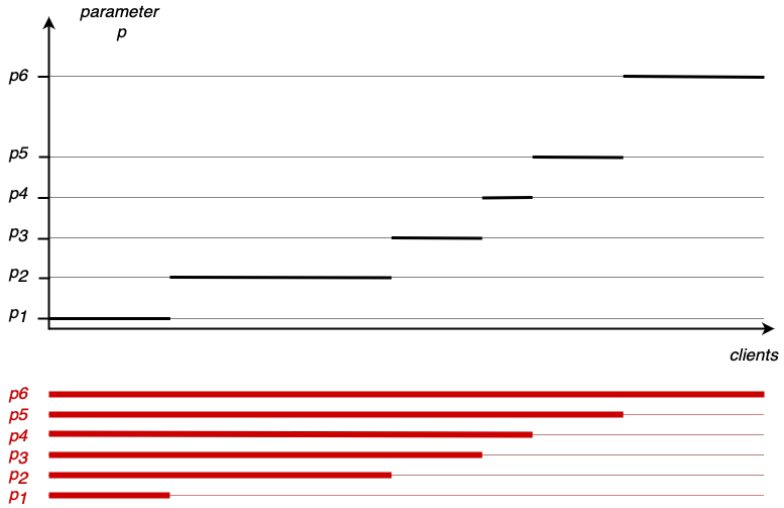



Figure 1.4: Here, the legend of the figure is the same as in Figure 1.3. The structure of the function for the parameter $p$ corresponds to the situation when $p$ represents a resource that is being delivered over time, and is spent only on servicing clients. In this scenario, the seller can serve only a certain number of clients until the next delivery of the resource.

incredibly fragile — even small changes in the prescribed arrival order bring them back to having good concentration. This motivates the study of instances in which the arrival order is not known ahead of time, but is instead generated uniformly randomly.

This is known as the *prophet secretary* problem, introduced by Esfandiari et al. in 2015 [EHLM17]. Algorithms for matroid prophet secretary against the prophet benchmark are well-studied, with a $1 - \frac{1}{e}$ approximation of the prophet given in [EHKS18]. However, there is little known in relation to the philosopher benchmark. In contrast to matroid Bayesian online selection being solvable in polynomial-time for all constant-rank matroids (see Subsection 1.2.1), the complexity of even the single-item case for prophet secretary is unknown. In 2023, Dütting et al. showed that there is a PTAS for the single-item case using a clever discretization and "front-loading" scheme [DGR+23]. In Chapter 4, we give a PTAS for matroid prophet secretary in all constant-rank matroids, subject to having a constant lower-bound on all probabilities in the instance.

The mechanisms given in [KW12] and [SC84] fundamentally require calculation of the prophet's expected revenue, or, in other words, the expected maximum basis weight of a matroid. In Chapter 5, we examine the computational difficulty of calculating this. We show that even in planar graphic matroids, computing the expected maximum weight basis is #P-hard, which in turn yields a hardness result for planar graphic Bayesian online

selection. On the positive side, we show that there is a fully polynomial-time randomized approximation scheme (FPRAS) for computing the expected maximum weight basis of any matroid. As a corollary of this result and [KW12], we find a $(2+\epsilon)$-approximation for matroid Bayesian online selection.

## 1.2 Overview of Techniques

### 1.2.1 PTAS for Left-to-Right Laminar Bayesian Online Selection

In the single-item prophet inequality problem, the philosopher's strategy can be computed in polynomial-time with a simple dynamic programming approach. Letting $f(i)$ denote the philosopher's expected revenue from buyers $i, i+1, \ldots, n$ (i.e., after rejecting buyers $1, 2, \ldots, i-1$), we can compute $f$ as

$$\begin{aligned} f(i) &= \mathbb{E}_{v_i \sim F_i}[\max\{v_i, f(i+1)\}] \qquad \forall\ 1 \le i \le n, \\ f(n+1) &= 0. \end{aligned}$$

Then, when considering buyer $i$'s offer $v_i$, the seller should accept buyer $i$'s offer if $v_i \ge f(i+1)$ and reject it otherwise. The function $f$ can be computed in time linear in the input size, so this is indeed a polynomial-time computation.

This approach can be generalized to work for matroid Bayesian online selection by encoding the set of sold items in our *state*. We let $f(i, S)$ denote the philosopher's expected revenue from buyers $i, i+1, \ldots, n$ given that the seller has already sold items $S \subseteq \{1, \ldots, i-1\}$. Recall that the seller is subject to selling a set that is independent in the matroid. We refer to independent sets as *feasible* and dependent sets as *infeasible*. Then we can compute $f$ as

$$\begin{aligned} f(i,S) &= \mathbb{E}_{v_i \sim F_i}[\max\{v_i + f(i+1, S\cup\{i\}), f(i+1,S)\}] && \forall\ 1 \le i \le n, S \text{ feasible}, \\ f(i,S) &= -\infty && \forall\ 1 \le i \le n+1, S \text{ infeasible}, \\ f(n+1,S) &= 0 && \forall\ S \text{ feasible}. \end{aligned}$$

The seller should accept buyer $i$'s offer at state $S$ when $v_i \ge f(i+1,S) - f(i+1, S\cup\{i\})$. The function $f$ can be computed in polynomial-time for constant-rank matroids using a simple dynamic programming approach. This is because the number of feasible sets $S$ is at most $O(n^r)$, where $r$ is the rank of the matroid. However, in general matroids, there can

be exponentially many "states" for $f$ to consider, making direct computation infeasible. Due to the inherent difficulty in understanding the behavior of the philosopher, most work done in relation to the philosopher benchmark proceeds by

1. breaking the given instance into smaller parts (often of constant rank),
2. solving the philosopher's problem optimally on each of these parts, and
3. showing that one can "combine" the optimal solutions to each part without incurring too much loss.

Indeed, the previous work on laminar Bayesian online selection by Anari et al. followed these steps exactly to give their polynomial-time approximation schemes. In Chapter 2, we give a PTAS for left-to-right laminar Bayesian online selection that generalizes both cases considered in [ANSS19]. True to form, our algorithm follows the steps outlined above.

We capture the optimal online policy for left-to-right laminar Bayesian online selection with an exponential-sized linear program introduced in [ANSS19]. While this linear program is too large to solve efficiently, we create a polynomial-sized relaxation by partitioning the sets of the laminar family into "big bins" and "small bins" based on their capacities. For each maximal small bin, we require the constraints from the exponential-sized linear program to hold exactly. But for each big bin, we only require its capacity constraints to "hold in expectation."

We show that by solving the linear program, we can efficiently obtain an online policy that is feasible for all small bins and has total expected revenue equal to the optimal value of the linear program. In fact, the obtained online policies are optimal for the maximal "small" bins subject to changes in the value distributions. The analysis about the above online policies goes generally along the lines of the analysis in [ANSS19]. One of our main technical contributions is showing that, with some pre-processing on capacities, big bins are unlikely to be violated by the obtained policies, see Lemma 4. To show this, we need concentration bounds on the number of items selected from each maximal small bin. The fact that the obtained policies act optimally on the maximal small bins does not guarantee us that selections of items are negatively correlated, see Figure 1.5, and not even that the selection of an item is negatively correlated with the number of items selected before it, see Figure 1.6. Thus, we cannot rely on the analysis from [ANSS19] and have to develop new tools. Nevertheless, we are able to show a more "global" version of negative dependence; we show that the number of items selected from each small bin is concentrated closely around the mean. Roughly speaking, we obtain this by showing that the selection of an item has a limited impact on the expected number of the items selected after it, see Lemma 7. This

gives us the building blocks needed for Chernoff type results, see Lemma 3, which we use to bound the probability of big bins being violated in Lemma 4.

These concentration results are only possible for restricted arrival orders, e.g., left-to-right arrival orders. We show in Theorem 4 that there exist laminar Bayesian online selection instances where the number of elements selected by the optimal policy is anti-concentrated. Due to the constructions done in the proof of Theorem 4, straightforward counting arguments show that given natural numbers $r$ and $n \in \Omega(r^2)$, for a randomly chosen laminar matroid over $n$ elements with rank $r$ and a randomly chosen order, asymptotically almost surely one can choose value distributions and capacities such that the resulting instance exhibits anti-concentration for the number of elements selected by the optimal policy. This is a significant roadblock on the way to giving a good approximation for laminar Bayesian online selection problems with no restriction on arrival order. Without strong concentration on the number of elements selected by the philosopher, the framework described above cannot work - further new ideas are needed to tackle the general problem.

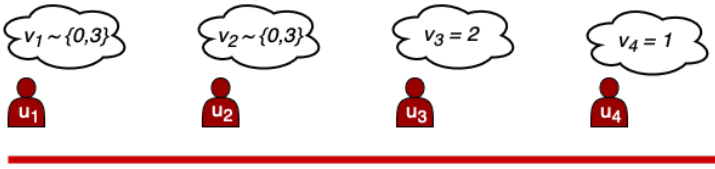



Figure 1.5: Here, the seller receives four clients $u_1$, $u_2$, $u_3$ and $u_4$ in the corresponding order, but can serve at most 2 clients. The distributions for the offers are as in the figure, e.g., $u_2$ offers 3 with probability 0.5 and otherwise offers 0, $u_4$ always offers 1. Let $X_i$, $i = 1, \ldots, 4$ be the event (and the corresponding indicator variable) that $u_i$ was served by the optimal online policy. We have $\mathbf{Pr}[X_3] = 3/4$, $\mathbf{Pr}[X_4] = 1/4$ while $\mathbf{Pr}[X_3 \wedge X_4] = 1/4$. Thus, we have $\mathbf{Cov}(X_3, X_4) = 1/16$.

### 1.2.2 Hardness of Matroid Bayesian Online Selection

To obtain PSPACE-hardness result for approximating Bayesian online selection for graphic matroids, we reduce from a stochastic variant of the maximum 2-satisfiability problem. We construct an instance with two special edges $wx_i, w\neg x_i$ for each variable $x_i$ in the given maximum 2-satisfiability instance. By carefully choosing values and adding some gadget edges, we can enforce that the optimal online algorithm selects exactly one of $wx_i$ and $w\neg x_i$, modelling the selection of truth values for $x_i$. Following this, we model each clause

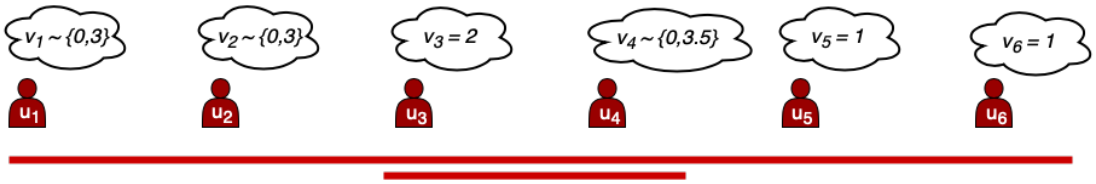



Figure 1.6: Here, the seller receives six clients $u_1, \ldots, u_6$ in the corresponding order, but can serve at most 3 clients. Moreover, among $u_3$ and $u_4$ the seller can serve at most one client. The distributions for the offers are as on the figure, e.g. $u_2$ makes offer 3 with probability 0.5 and otherwise makes offer 0, $u_5$ always offers 1. Let $X_i$, $i = 1, \ldots, 6$ be the event (and the corresponding indicator variable) that $u_i$ was served by the optimal online policy. We have $\mathbf{Pr}[X_1] = \mathbf{Pr}[X_2] = 1/2$, $\mathbf{Pr}[X_3] = 1/4$, $\mathbf{Pr}[X_4] = 3/8$, $\mathbf{Pr}[X_1 \wedge X_4] = \mathbf{Pr}[X_2 \wedge X_4] = 1/4$ and $\mathbf{Pr}[X_3 \wedge X_4] = 0$. Thus, we have $\mathbf{Cov}(X_1 + X_2 + X_3, X_4) = 1/32$.

$(\ell_i \vee \ell_j)$ with the edge $\neg\ell_i\neg\ell_j$. If all other clause edges are ignored, this edge is selectable if and only if either $w\ell_i$ or $w\ell_j$ was selected by the algorithm. By adjusting clause edges to have zero value "most of the time", we are able to characterize the performance of the optimal online algorithm by how many clause edges are selectable. This allows us to translate a near-optimal solution for graphic Bayesian online selection to a near-optimal solution for maximum 2-satisfiability, obtaining NP-hardness of approximation. To obtain our PSPACE-hardness result, we encode additional rules on which $wx_i, w\neg x_i$ edges may be selected by the optimal online algorithm. See Section 3.1 for further details.

In Section 3.2, we show that essentially the same hardness construction can also be modelled with transversal matroids. This proves that transversal Bayesian online selection is also PSPACE-hard to approximate to some fixed constant.

We round off our hardness results in Section 3.3 by showing that it is NP-hard to approximate co-graphic Bayesian online selection to some fixed constant $\alpha \in (0, 1)$. To show this, we reduce from the *Tree Augmentation Problem*, the problem of finding a minimum cost set of edges to add to a tree to make it two edge connected. The reduction begins with all possible augmenting edges arriving, each with a deterministic value equal to their cost. Next, each edge of the tree arrives, taking on high value with low probability, and zero value otherwise. We show that optimal behavior in this instance involves choosing the cheapest set of edges to *not* select among the augmenting edges, subject to these edges making the input tree two edge connected when added.

### 1.2.3 PTAS for Constant-Rank Prophet Secretary

We first perform a discretization step that allows us to assume there are a constant number of distinct possible support values $w_1 \geq w_2 \geq \ldots \geq w_k$ with minor loss.

Suppose we are in the single-item case. If there is a "sufficient" number of items that could take on value $w_1$, then with probability nearly 1, one of them will actually have value $w_1$. Thus, we can assume with small error that the philosopher's expected revenue is $w_1$. But suppose there are not enough elements that could take on value $w_1$ to assume this. Then if there are a "sufficient" number of items that could take on value $w_2$, then with probability nearly 1, one of them will actually have value $w_2$ *and* arrive after all elements that could take on value $w_1$. So we can assume in this case that the philosopher first waits to see if any element has value $w_1$, and if not, will always be able to take $w_2$.

This process can be continued inductively — suppose that $w_i$ is the highest weight which is deemed to have a "sufficient" number of elements that could take on value $w_i$. Then there is only a small set of elements that could possibly take on any values in $\{w_1, w_2, \ldots, w_{i-1}\}$, call this set $S$. Then an algorithm that ignores all elements outside $S$, except for assuming that there is a guaranteed option to earn $w_i$ at the very end, will perform very similarly to the philosopher. With the right definition of "sufficient", we can assume that $S$ has constant size and so this algorithm runs in constant time.

In order to generalize to the matroid case, we instead consider "concentration" on flats. For any flat $F$ and value $w_i$, if there is a "sufficient concentration" of elements with support value $w_i$ in $F$, i.e., there are many disjoint bases of $F$ with support value $w_i$, then the optimal online algorithm is very unlikely to accept any offer with value smaller than $w_i$ in $F$. Therefore, we can ignore all values smaller than $w_i$ in $F$ with minor loss. Roughly speaking, our algorithm proceeds by examining weights in decreasing order, and for each, contracting a flat in which we ignore elements of lower weight. By selecting this flat appropriately, the rest of the matroid will contain only a constant number of elements of our considered weight, allowing us to keep track of them exactly with a dynamic programming type solution.

## 1.3 Overview of Related Work

This thesis is primarily concerned with constraints defined by *matroids*. Due to the fact that we work almost entirely with special cases of matroids, we rarely make use of the rich theory developed about matroids over the past decades. The main property we use is that

the simple greedy algorithm is optimal for finding a maximum weight basis in a weighted matroid. For further reading, a summary of many interesting results relating to matroids and their structure can be found in the book by Oxley [Oxl06].

In recent years, there has been an increasing amount of attention given to prophet inequalities and related problems. This increased attention has been spurred on primarily by the connections between prophet inequalities and mechanism design. The connection between these areas is deep and has been expanded on in numerous papers, e.g., [HKS07, BHK+24, AKW19], along with the excellent survey by Lucier [Luc17].

Much of this recent work has been dedicated to the study of prophet inequalities under combinatorial constraints. Beginning with uniform matroid constraints [HKS07, Ala11], there has been exploration of general matroid constraints [KW12], knapsack constraints [FSZ16], matching constraints [AHL12, GW19, EFGT20], general downwards-closed constraints [RS17, Rub16], and many more.

There have been numerous variations of the prophet inequality that modify things besides constraints on sets of items which can be sold together. Some examples include the aforementioned prophet secretary problem in which the arrival order of elements is random [EHKS18, EHLM17, CSZ21, ACK18], IID prophet inequalities, where every element has its value drawn independently from the same distribution [AEE+17, CFH+17, HK82], single sample prophet inequalities [RWW19, AKW14], and non-adaptive prophet inequalities [CGKM20, PS23].

While the prophet benchmark has received the bulk of attention, work considering the philosopher benchmark has started to pick up steam. Online stochastic weighted bipartite matching has been extensively studied in the last several years, with a series of improved approximation factors given in [PPSW21, BDML22, NSW23, BDP+24]. [ASZ20] shows that even in the single-item prophet inequality problem, it is NP-hard to select the best order for the philosopher to observe elements in. Additional approximations to the online optimum in stochastic and Bayesian online selection problems have been given in a variety of settings [FNS22, SS21, FNS21, FLX18].

## 1.4 Preliminaries

In this thesis, we consider structural constraints defined by matroids. Each matroid $\mathcal{M}$ is defined by a ground set $U$ and a collection of subsets of $U$, which are called *independent sets*.

We will work mainly with two types of matroids: laminar and graphic. We say that a matroid $\mathcal{M}$ over a ground set $U$ is *laminar* if there is some laminar family of sets $\mathcal{L}$ over $U$ and a capacity function $c : \mathcal{L} \to \mathbb{N}$ such that a set $I$ is independent if and only if $|I \cap A| \leq c(A)$ for all $A \in \mathcal{L}$. In this case, we write $\mathcal{M} = (U, \mathcal{L}, c)$. Given a laminar matroid $\mathcal{M} = (U, \mathcal{L}, c)$, we call the sets in $\mathcal{L}$ *bins*. The *depth* of a bin $A \in \mathcal{L}$ is defined as $|\{B : A \subseteq B, B \in \mathcal{L}\}|$.

We say that a matroid $\mathcal{M}$ over a ground set $U := E$ is *graphic* if there is some graph $G = (V, E)$ such that a set $I \subseteq E$ is independent if and only if $(V, I)$ has no cycles. For further reading about matroids and their properties, we refer to the excellent book by Oxley [Oxl06].

From now on, we also refer to the seller as a *gambler* and to the items as *elements*. We also interchangeably use the words gain and revenue. In the matroid Bayesian online selection problem, the gambler is given the matroid $\mathcal{M}$ over a ground set $U = \{u_1, \ldots, u_n\}$, an arrival order of the elements in $U$, and for each element $u_i \in U$, a distribution $F_i$ for its value $v_i$. Over the course of the game, the gambler maintains an independent set $I$ which starts as the empty set. The set $I$ consists of all elements selected by the gambler at the given timestamp.

Elements arrive one by one in the given order. When $u_i$ arrives, the value $v_i \sim F_i$ is drawn from its distribution independently from values of other elements and presented to the gambler. The gambler may then choose to either select the element as long as independence is maintained, updating $I \leftarrow I \cup \{u_i\}$ and gaining the associated revenue $v_i$, or reject the element, gaining no revenue. This decision is final and may not be changed later in the game.

The goal of the gambler is to gain as much revenue as possible over the course of the game. Let $OPT_{\mathcal{M}}$ indicate the maximum expected revenue that the gambler can achieve for the instance $\mathcal{M}$. Here and elsewhere, we abuse notation and associate the matroid Bayesian online selection instance to the underlying matroid $\mathcal{M}$ when the value distributions and arrival order are not relevant or are clear from the context.

The Laminar Bayesian Online Selection (`LBOS`) problem is the matroid Bayesian online selection problem restricted to laminar matroids. The Graphic Bayesian Online Selection (`GBOS`) problem is the matroid Bayesian online selection problem restricted to graphic matroids.

# Chapter 2

# PTAS for Left-to-Right Laminar Bayesian Online Selection

Past work on the LBOS problem has focused on special cases. Namely, [ANSS19] gives a PTAS in the constant-depth case. They also give a PTAS for nested laminar matroids — a special case where both the structure of the laminar matroid and the arrival order are restricted.

In this chapter, we generalize both of these results by providing a PTAS for LBOS instances that satisfy a natural "left-to-right" restriction on their arrival order. In particular, once elements from some bin of the laminar matroid begin arriving, they may not stop until all elements from that bin have arrived. This is a near-generalization of the arrival order required for nested laminar matroids, as nested laminar matroids are equipped with many left-to-right orders joined by a single global constraint. In Section 2.5 we show that such a strict arrival order is not necessary — a PTAS can still be found when some bins violate this arrival order constraint, generalizing both results from [ANSS19].

To obtain this result, we first pre-process our laminar matroid to obtain a nearly-equivalent instance with logarithmic depth. Following this, we apply a similar linear program and rounding scheme as used in [ANSS19], along with a novel analysis of concentration, to obtain our PTAS.

## 2.1 Preliminaries

Recall that a laminar matroid

$$\begin{aligned} f(i,S) &= \mathbb{E}_{v_i \sim F_i}[\max\{v_i + f(i+1, S \cup \{i\}), f(i+1,S)\}] && \forall\ 1 \le i \le n, S \text{ feasible}, \\ f(i,S) &= -\infty && \forall\ 1 \le i \le n+1, S \text{ infeasible}, \\ f(n+1,S) &= 0 && \forall\ S \text{ feasible}. \end{aligned}$$

is a matroid $\mathcal{M} = (U, \mathcal{L}, c)$ over a ground set $U$, with a set $I \subseteq U$ being independent if and only if $|I \cap A| \le c(A)$ for all $A \in \mathcal{L}$. Here, $\mathcal{L}$ must be a laminar family of sets.

**Definition 1.** *Given a laminar matroid $\mathcal{M} = (U, \mathcal{L}, c)$ and an ordering $u_1, u_2, \ldots, u_n$ of the elements $U$, we say that this is a* left-to-right ordering *if the following condition holds:*

> *For every bin $A \in \mathcal{L}$ of the laminar family, if $u_i, u_j \in A$ for $i < j$, then for every $k \in [i, j]$, we must have $u_k \in A$.*

Left-to-right orderings capture the rule that once elements from some bin start to arrive, they must not stop until they have all arrived. We say that an LBOS *instance has a left-to-right arrival* order (or equivalently, it is a *left-to-right LBOS instance*) if the elements arrive according to a left-to-right ordering.

## 2.2 Warm-Up: QPTAS

In this section, we demonstrate our depth pre-processing step and show that it immediately yields a Quasi-Polynomial Time Approximation Scheme (QPTAS).

**Theorem 1.** *There is a QPTAS for left-to-right Laminar Bayesian Online Selection.*

To design both our QPTAS and, in the next section, our PTAS, we rely on the concept of states. A *state* has an entry for each considered bin $A$ in $\mathcal{L}$ equal to $c(A)$ minus the number of currently selected elements from $A$. Informally, a state represents the remaining capacities of the bins under consideration.

A straightforward (but exponential-time) exact algorithm for LBOS can be obtained via a dynamic programming solution which calculates the expected value of the philosopher on elements $\{u_i, \ldots, u_n\}$ given a timestamp $i$ and the state of all bins.

In an LBOS instance, given a timestamp, let us call a bin $A$ *active* if at least one element of $A$ already arrived and there are still some elements of $A$ left to arrive. Critically, in a left-to-right arrival order, a bin $A$ can only be active at timestamp $i$ if $u_i \in A$. The next lemma is based on the fact that in an LBOS instance, we only need to keep track of states for active bins, as the remaining capacity of inactive bins are either irrelevant or equal to their initial capacity.

**Lemma 1.** *Let $\mathcal{M} = (U, \mathcal{L}, c)$ be a left-to-right LBOS instance with depth at most $L$. Then the philosopher's expected revenue in $\mathcal{M}$ can be computed in $n^{\mathcal{O}(L)}$ time.*

*Proof.* Note that at any timestamp, there are at most $L$ active bins due to the left-to-right ordering and definition of depth. This computation can then be done with a dynamic programming algorithm that computes the expected revenue of the philosopher from elements $\{u_i, \ldots, u_n\}$ given a timestamp $i$ and the state of all bins active at time $i$.

The number of states is at most $n \cdot (n+1)^L$ as there are $n$ timestamps and the remaining capacity of each active bin is an integer in $[0, n]$.

□

**Corollary 1.** *The optimal expected revenue and an optimal policy of the gambler in a constant-depth left-to-right LMBS instance can be computed in polynomial time.*

Given Lemma 1, all that remains is to show how we can preprocess an LBOS instance $\mathcal{M}$ to a nearly-equivalent instance with logarithmic depth.

**Lemma 2.** *Let $\mathcal{M} = (U, \mathcal{L}, c)$ be a left-to-right LBOS instance and $\alpha$ be in $(0, 1) \cap \mathbb{Q}$. We can, in polynomial time, construct an LBOS instance $\mathcal{M}' = (U, \mathcal{L}', c')$ such that:*

- $\mathcal{L}' \subseteq \mathcal{L}$
- *for all $A$, $B \in \mathcal{L}'$ with $A \subsetneq B$, we have*

$$c'(A) \leq \lceil \alpha \cdot c'(B) \rceil \tag{SEP}$$

- *all independent sets in $\mathcal{M}'$ are also independent in $\mathcal{M}$*
- $\alpha \cdot OPT_{\mathcal{M}} \leq OPT_{\mathcal{M}'} \leq OPT_{\mathcal{M}}$
- *$\mathcal{M}'$ is also a left-to-right instance.*

*Proof.* For the construction, let us start with empty $\mathcal{L}'$ and process the bins in $\mathcal{L}$ in a non-increasing order of their cardinality.

Let $A \in \mathcal{L}$ be the currently processed bin. If $A$ is not contained in any other bins from $\mathcal{L}'$, add $A$ to $\mathcal{L}'$ and set $c'(A) := c(A)$. Otherwise, let $B \in \mathcal{L}'$ be the inclusion-wise minimal bin in $\mathcal{L}'$ containing $A$. If $c(A) \geq c'(B)$, we move to the next unprocessed bin from $\mathcal{L}$. Otherwise, we add $A$ to $\mathcal{L}'$ and set $c'(A) := \min\{c(A), \lceil \alpha \cdot c'(B) \rceil\}$. This transformation yields the desired separation property (SEP).

Note that any independent set in $\mathcal{M}'$ is also independent in $\mathcal{M}$. Indeed, if $A$ is in $\mathcal{L}'$, then we have $c'(A) \leq c(A)$. If $A$ is in $\mathcal{L} \setminus \mathcal{L}'$, then $c(A) \leq c'(B)$ where $B \in \mathcal{L}'$ is the inclusion-wise minimal bin in $\mathcal{L}'$ containing $A$. So, we have $OPT_{\mathcal{M}'} \leq OPT_{\mathcal{M}}$.

The fact that $\mathcal{M}'$ is a left-to-right instance follows directly from the fact that $\mathcal{L}' \subseteq \mathcal{L}$.

To see why $OPT_{\mathcal{M}'} \geq \alpha \cdot OPT_{\mathcal{M}}$, consider a strategy for $\mathcal{M}'$ that follows an optimal strategy for $\mathcal{M}$, but selects only a portion of the elements. Denote $\alpha = \frac{p}{q}$ for positive integer $p, q$. Roughly speaking, the algorithm for $\mathcal{M}'$ partitions the items that the optimal algorithm for $\mathcal{M}$ selects into intervals of length $q$. Then the algorithm for $\mathcal{M}'$ will select $p$ "evenly spaced" elements from each interval.

More precisely, let $S \subseteq [0, q-1]$ be a set with size $|S| = p$ and "optimal spacing". That is, $S$ is built greedily in order of $i = 0, 1, \ldots, q-1$ by selecting $i$ if $|S| \leq \alpha \cdot i$ and skipping $i$ otherwise. Note that for any interval $[i, j] \subseteq [0, q-1]$, we have that $|[i, j] \cap S| \leq \lceil \alpha \cdot (j - i + 1) \rceil$. That is, any sub-interval has concentration at most $\alpha$ subject to rounding up.

The algorithm for $\mathcal{M}'$ proceeds by selecting a uniformly random integer offset $o \in [0, q-1]$. For an index $i$, let $\sigma(i)$ denote the smallest non-negative integer such that $\sigma(i)$ is congruent to $i$ modulo $q$. Then the algorithm for $\mathcal{M}'$ rejects all items that the optimal algorithm for $\mathcal{M}$ rejects, and accepts the $i$th element that the optimal algorithm for $\mathcal{M}$ selects if and only if $\sigma(i - o) \in S$.

This strategy is feasible due to the left-to-right arrival order on $\mathcal{M}$ and $\mathcal{M}'$. Consider a bin $B \in \mathcal{L}'$ with original capacity $c(B)$. The elements selected by the optimal algorithm for $\mathcal{M}$ that lie in $B$ occur in consecutive order among all selected elements. Therefore, due to how $S$ was constructed, the algorithm for $\mathcal{M}'$ will select at most $\lceil \alpha \cdot c(B) \rceil \leq c'(B)$ elements from $B$. Thus, the algorithm for $\mathcal{M}'$ is feasible.

Finally, note that due to the random selection of $o$, this algorithm for $\mathcal{M}'$ has a $\alpha$ chance of selecting any element that $\mathcal{M}$ selects. Thus, the expected revenue of the algorithm for $\mathcal{M}'$ is $\alpha \cdot OPT_{\mathcal{M}}$.

□

*Proof of Theorem 1.* Given a left-to-right LBOS instance $\mathcal{M}$, we can use Lemma 2 with $\alpha = 1 - \epsilon$ to get an instance $\mathcal{M}'$. By (SEP) with $\alpha = 1 - \epsilon$, the depth of $\mathcal{M}'$ is $\mathcal{O}(\log_{1/(1-\epsilon)}(n))$. Now we can use Lemma 1 to compute the optimal expected revenue and policy for $\mathcal{M}'$ in $n^{\mathcal{O}(\log_{1/(1-\epsilon)}(n))} \leq n^{\mathcal{O}(\log(n)/\epsilon)}$ time. By Lemma 2, we obtain a $(1-\epsilon)$-approximation to the philosopher's expected revenue for $\mathcal{M}$. □

## 2.3 Linear Programming Formulation and Rounding Algorithm

A naive dynamic programming algorithm for LBOS can be written as in the proof of Lemma 1. However, the number of considered states makes this approach not viable. To circumvent this, we build on a linear programming formulation from [ANSS19]. The linear program from [ANSS19] encodes dynamic programming ideas as in Lemma 1. Their linear program allows us to construct a relaxation by decomposing a given instance of LBOS into tractable parts and concentrating on optimal policies for each of these tractable parts. For the sake of completeness, we present the linear program and relevant results from [ANSS19] in this section.

### 2.3.1 Small and Big Bins

To decompose an instance of LBOS into tractable parts, we partition $\mathcal{L}$ into "big" and "small" bins based on their capacities. If a bin is small, we use a linear program based on dynamic programming to guarantee its "feasibility." If a bin is big, we enforce that its "feasibility" is guaranteed in expectation.

Consider a threshold $K \in \mathbb{N}$. Call a bin $A \in \mathcal{L}$ *big* if $c(A) \geq K$, and *small* otherwise. Let $\mathscr{L} \subseteq \mathcal{L}$ indicate the set of big bins and $\mathscr{S} \subseteq \mathcal{L}$ indicate the set of inclusion-wise maximal small bins.

Without loss of generality, we assume that each element of $U$ is contained in some small bin. If we have an element not contained in any small bin, then we can add a bin containing only this element. So, we can assume that $\mathscr{S}$ partitions $U$.

### 2.3.2 Feasibility on Small Bins

Let us consider a maximal small bin $B \in \mathscr{S}$. Let $\mathcal{L}^B = \{B' \in \mathcal{L} : B' \subseteq B\}$ be the set of bins contained in $B$. To capture states with respect to $B$ at each timestamp, we use $\mathcal{S}^B \subseteq \mathbb{Z}^{\mathcal{L}^B}$ to indicate the set of feasible states. Again, each vector $\vec{s} \in \mathcal{S}^B$ has an entry for every $A \in \mathcal{L}^B$ which represents $c(A)$ minus the number of selected elements from $A$. Note that $|\mathcal{S}^B|$ is at most $n^{\mathcal{O}(c(B))}$ because no more than $c(B)$ elements can be selected from $B$ by an online policy.

Let $\vec{d_t} \in \{0,1\}^{\mathcal{L}^B}$ be the indicator vector for the bins in $\mathcal{L}^B$ containing the element $u_t$. Let us define the set of *forbidden neighbouring states*

$$\partial\mathcal{S}^B := \{\vec{f} \in \mathbb{Z}^{\mathcal{L}^B} \setminus \mathcal{S}^B : \text{there are } u_t \in B \text{ and } \vec{s} \in \mathcal{S}^B \text{ such that } \vec{s} = \vec{f} + \vec{d_t}\}\,.$$

In other words, $\partial\mathcal{S}^B$ contains the set of all states that can be reached by an online policy by selecting an infeasible item while in a feasible state, in addition to some irrelevant unreachable states.

We use *allocation variables* $\mathcal{X}_t(\vec{s}, v)$, which represent the probability that the gambler accepts the $t$th item $u_t$ and the state upon its arrival is $\vec{s}$, conditioned on $v$ being the realized value of $v_t$. We then use $\mathcal{X}_t(v)$ to represent the conditional probability that we accept $u_t$, conditioned on $v$ being the realized value of $v_t$. The *state variables* $\mathcal{Y}_t(\vec{s})$ represent the probability that the gambler is at state $\vec{s}$ upon the arrival of $u_t$.

Now we introduce a polytope $\mathcal{P}^B$ to capture transitions between states through an application of an online policy on the bin $B$. The polytope $\mathcal{P}^B$ is defined by the following constraints:

$$\mathcal{X}_t(v) = \sum_{\vec{s} \in \mathcal{S}^B} \mathcal{X}_t(\vec{s}, v) \qquad \forall v, u_t \in B \tag{2.1}$$

$$0 \le \mathcal{X}_t(\vec{s}, v) \le \mathcal{Y}_t(\vec{s}) \qquad \forall v, \vec{s} \in \mathcal{S}^B, u_t \in B \tag{2.2}$$

$$\mathcal{Y}_{t+1}(\vec{s}) = \mathcal{Y}_t(\vec{s}) - \mathbb{E}_{v_t}[\mathcal{X}_t(\vec{s}, v_t)] + \mathbb{E}_{v_t}[\mathcal{X}_t(\vec{s} + \vec{d_t}, v_t)] \qquad \forall \vec{s} \in \mathcal{S}^B, u_t, u_{t+1} \in B \tag{2.3}$$

$$\mathcal{Y}_{t_0}([c(A)]_{A \in \mathcal{L}^B}) = 1 \qquad t_0 = \min\{t : u_t \in B\} \tag{2.4}$$

$$\mathcal{Y}_t(\vec{s}) = 0 \qquad \forall \vec{s} \in \partial\mathcal{S}^B, u_t \in B\,. \tag{2.5}$$

Constraint (2.1) ensures agreement between $\mathcal{X}_t(v)$ and $\mathcal{X}_t(\vec{s}, v)$ variables. Constraint (2.2) ensures that the probability of selecting some item in a given state cannot exceed the probability of reaching that state. Constraint (2.3) updates $\mathcal{Y}_t(\vec{s})$ variables by examining the transition probabilities according to $\mathcal{X}_t(\vec{s}, v)$ variables. Constraint (2.4) ensures a

proper beginning state for $\mathcal{Y}_t(\vec{s})$ variables. Constraint (2.5) ensures that the resulting policy does not violate feasibility.

One can see that any online policy for the restriction of $\mathcal{M}$ to $B$ induces a feasible point in $\mathcal{P}^B$. Let us show that the opposite also holds.

**Proposition 1** (Proposition 2.1 from [ANSS19]). *Given a bin $B \in \mathscr{S}$ and a point $\{\mathcal{X}_t(\vec{s}, v), \mathcal{X}_t(v), \mathcal{Y}_t(\vec{s})\} \in \mathcal{P}^B$, there is an online policy for the restriction of $\mathcal{M}$ on $B$ which guarantees the expected gambler's revenue to be $\sum_{u_t \in B} \mathbb{E}_{v_t}[v_t \cdot \mathcal{X}_t(v_t)]$ and the expected number of selected elements to be $\sum_{u_t \in B} \mathbb{E}_{v_t}[\mathcal{X}_t(v_t)]$. Moreover, this policy can be found in polynomial time.*

*Proof.* Let us outline the desired policy. Let us assume that the gambler is at the state $\vec{s}$ at the timestamp $t$ and $v$ is the realized value of $v_t$. If $\mathcal{Y}_t(\vec{s})$ is nonzero, the gambler accepts the element $u_t$ with probability $\mathcal{X}_t(\vec{s}, v)/\mathcal{Y}_t(\vec{s})$. Otherwise, the gambler rejects the element $u_t$. It is straightforward to see that this policy is feasible and guarantees the required revenue and the expected number of selected elements. □

### 2.3.3 Feasibility on Big Bins

For each maximal small bin $B \in \mathscr{S}$, we introduce a variable $\mathcal{N}_B$ to represent an upper bound on the expected number of elements selected from $B$. Using these variables, we construct our final linear program.

$$
\begin{aligned}
&\text{maximize} && \sum_{t=1}^{n} \mathbb{E}_{v_t}[v_t \cdot \mathcal{X}_t(v_t)] && && (LP)\\
&\text{subject to} && \sum_{A \in \mathscr{S} : A \subseteq B} \mathcal{N}_A \le c(B) && \forall B \in \mathscr{L}\\
& && \sum_{u_t \in B} \mathbb{E}_{v_t}[\mathcal{X}_t(v_t)] \le \mathcal{N}_B && \forall B \in \mathscr{S}\\
& && \{\mathcal{X}_t(\vec{s}, v), \mathcal{X}_t(v), \mathcal{Y}_t(\vec{s})\} \in \mathcal{P}^B && \forall B \in \mathscr{S}
\end{aligned}
$$

Note that in ($LP$) the first type of constraints are global ex-ante constraints. They enforce the "feasibility" of big bins in expectation. Note that each item is contained in a maximal small bin by assumption, so these constraints truly keep track of all items in the big bin. The second type of constraints enforces that for each $B \in \mathscr{S}$, the variable $\mathcal{N}_B$ is a correct upper bound on the expected number of selected elements. Finally, the third type of constraints guarantees "feasibility" on all small bins. One can see that any online policy induces a feasible solution for ($LP$).

### 2.3.4 Algorithm

Now we are ready to state the algorithm. We pre-processes an input LBOS instance as in Lemma 2 to guarantee the separation property (SEP) with $\alpha = (1 - \epsilon)$. After that we divide the bins into small and big as in Section 2.3.1. We decrease the capacities for big bins in order to introduce "slack" for our online policy to satisfy their constraints. The capacities of small bins are satisfied by our online policy due to the constraints in the linear program ($LP$).

**Algorithm 1:** ALG($\mathcal{M} = (U, \mathcal{L}, c)$, $K$, $\epsilon$)

1 Obtain $\mathcal{M}' = (U, \mathcal{L}', c')$ from $\mathcal{M} = (U, \mathcal{L}, c)$ as in Lemma 2 based on $\alpha = (1 - \epsilon)$ ;
2 Compute maximal small bins $\mathscr{S}$ and big bins $\mathscr{L}$ for $\mathcal{M}'$ as in Section 2.3.1 based on $K$ ;
3 Obtain $\mathcal{M}'' = (U, \mathcal{L}', c'')$ from $\mathcal{M}' = (U, \mathcal{L}', c')$ by setting $c''(B) := (1 - \epsilon) \cdot c'(B)$ for $B \in \mathscr{L}$; and $c''(B) := c'(B)$ for $B \in \mathcal{L}' \setminus \mathscr{L}$ ;
4 Compute the optimal solution $\{\mathcal{X}^\star_t(\vec{s}, v_t), \mathcal{X}^\star_t(v_t), \mathcal{Y}^*_t(\vec{s}), \mathcal{N}^*_A\}$ for the linear program ($LP$) with respect to $\mathcal{M}''$ ;
5 Extract an online policy from $\{\mathcal{X}^\star_t(\vec{s}, v_t), \mathcal{X}^\star_t(v_t), \mathcal{Y}^*_t(\vec{s})\}$ for each $B \in \mathscr{S}$ as in Proposition 1 ;
6 Independently run the obtained online policies on $B \in \mathscr{S}$. Select an element as long as the obtained policy suggests selecting it and the selection is feasible with respect to $\mathcal{M}'$ ;

Note that this is a polynomial-time algorithm because the size of small bins are bounded above by a constant, and thus the size of the LP solved in step 4 is polynomial.

## 2.4 Analysis

In this section, we estimate the approximation guarantees achieved by Algorithm 1 with respect to an optimal online policy.

From now on, we assume for simplicity that each distribution is non-atomic — that is, every set with measure 0 has 0 probability mass. This simplifies the analysis of the optimal online policy, because for non-atomic distributions the optimal online policy is both unique and deterministic. A similar analysis applies to atomic distributions — the randomness in optimal online policies can be accounted for by considering small dispersals of the values in each distribution. As the dispersals grow smaller, the behavior of the optimal online

policy for the non-atomic case becomes the behavior of the optimal online policy for the atomic case.

### 2.4.1 Losses from Discarding Elements in Step 6 of Algorithm 1

The key challenge of analyzing Algorithm 1 is to estimate the expected value of elements that were not selected in Step 6 due to the feasibility restrictions of $\mathcal{M}'$. Lemma 4 shows that each element has a rather small probability of being not selected due to the feasibility restrictions of $\mathcal{M}'$. To show this, we first need the following technical result about the concentration on the number of elements chosen in each small bin. We defer its proof to Section 2.4.5.

**Lemma 3.** *Given a maximal small bin $B \in \mathscr{S}$ and an element $u_t \in B$, let $X_t$ be the event that the online policies computed in Step 5 of Algorithm 1 suggest selecting $u_t$. Then, for every $\alpha > 0$ we have that*

$$\mathbb{E}[e^{\alpha \cdot X_B}] \leq e^{(e^\alpha - 1) \cdot \mathbb{E}[X_B]},$$

*where $X_B := \sum_{u_t \in B} X_t$.*

**Lemma 4.** *In left-to-right* `LBOS`*, for each $u_t \in U$, the probability that upon the arrival of $u_t$ the online policies computed in Step 5 suggest selecting $u_t$ but it is not feasible with respect to $\mathcal{M}'$ is at most $3/(K \cdot \epsilon^3)$.*

*Proof.* For each item $u_i \in U$, let $X_i$ be the event that the online policies computed in Step 5 suggest selecting $u_i$. Given a bin $B \in \mathcal{L}'$, let $X_B := \sum_{u_i \in B} X_i$.

By the design of Algorithm 1, for every small bin $B \in \mathscr{S}$, the sum $\sum_{u_i \in B} X_i$ is always at most $c'(B)$. Now, consider any big bin $B \in \mathscr{L}$ with $u_t \in B$. Let us bound the probability that $X_B$ exceeds $c'(B)$. First, we note that $\mathbb{E}[X_B] \leq (1-\epsilon) \cdot c'(B)$ due to the structure of (*LP*) in Step 4 and $c''(B) \leq (1-\epsilon) \cdot c'(B)$. By Lemma 3, we have strong bounds on the concentration of $X_A$ for every maximal small bin $A \in \mathscr{S}$ with $A \subseteq B$. For every $\alpha > 0$, we have $\mathbb{E}[e^{\alpha X_A}] \leq e^{(e^\alpha - 1) \cdot \mathbb{E}[X_A]}$.

Note that the suggestions of online policies from Step 5 on different maximal small bins $A \in \mathscr{S}$ are independent. Since we assumed that every element is contained in some maximal small bin we have that

$$\mathbb{E}[e^{\alpha \cdot X_B}] = \mathbb{E}[e^{\alpha \cdot \sum_{A \in \mathscr{S},\, A \subseteq B} X_A}] = \prod_{A \in \mathscr{S},\, A \subseteq B} \mathbb{E}[e^{\alpha \cdot X_A}] \leq e^{(e^\alpha - 1) \cdot \sum_{A \in \mathscr{S},\, A \subseteq B} \mathbb{E}[X_A]} \leq e^{(e^\alpha - 1) \cdot (1-\epsilon) \cdot c'(B)}.$$

Hence, by applying Markov's inequality as used in showing Chernoff type results, we have

$$\mathbf{Pr}[X_B > c'(B)] = \mathbf{Pr}[e^{\alpha \cdot X_B} > e^{\alpha \cdot c'(B)}] \leq e^{(e^{\alpha}-1)\cdot(1-\epsilon)\cdot c'(B)} / e^{\alpha \cdot c'(B)} = e^{((e^{\alpha}-1)(1-\epsilon)-\alpha)\cdot c'(B)}$$

and by selecting $\alpha = \ln(1+\epsilon)$ and by $\ln(1+\epsilon) \geq \epsilon/(1+\epsilon/2)$

$$\mathbf{Pr}[X_B > c'(B)] \leq e^{(\epsilon\cdot(1-\epsilon)-\ln(1+\epsilon))\cdot c'(B)} \leq e^{-\frac{\epsilon^2}{(2+\epsilon)}\cdot c'(B)}\,.$$

Recall that for any two big bins $B$, $B' \in \mathscr{L}$ such that $B \subsetneq B'$ we have $c'(B) \leq (1-\epsilon)\cdot c'(B')$. Recall also that for any big bin $B \in \mathscr{L}$ we have $c'(B) \geq K$. Thus, using our previous estimates, we have

$$\sum_{\substack{B\in\mathscr{L}\\ u_t\in B}} \mathbf{Pr}[X_B > c'(B)] \leq \sum_{j=1}^{\infty} e^{-\frac{\epsilon^2}{(2+\epsilon)}\cdot K/(1-\epsilon)^j} \leq \sum_{j=1}^{\infty} \frac{2+\epsilon}{K\cdot\epsilon^2}(1-\epsilon)^j \leq \frac{3}{K\cdot\epsilon^3}\,,$$

finishing the proof. □

### 2.4.2 Losses from Transformations in Steps 1-3 of Algorithm 1

There are two steps in Algorithm 1 where we transform the original LBOS problem: Step 1 and Step 3. The next lemma captures how the optimal revenue changes between these transformations. We are also interested in the revenue estimations obtained through (*LP*). So, let us define $LP_{\mathcal{M}'}$ and $LP_{\mathcal{M}''}$ to indicate the optimal objective values of (*LP*) with respect to $\mathcal{M}'$ and $\mathcal{M}''$, respectively.

**Lemma 5.** *In Algorithm 1, the following inequality holds*

$$LP_{\mathcal{M}''} \geq (1-\epsilon)^2 \cdot OPT_{\mathcal{M}}\,.$$

*Proof.* By Lemma 2, we have

$$OPT_{\mathcal{M}'} \geq (1-\epsilon)\cdot OPT_{\mathcal{M}}.$$

By definition of (*LP*) we have $LP_{\mathcal{M}'} \geq OPT_{\mathcal{M}'}$. To finish the proof, note that any feasible point in the polytope (*LP*) for $\mathcal{M}'$ becomes feasible in (*LP*) for $\mathcal{M}''$ through a multiplication by a factor of $(1-\epsilon)$. Thus, we obtain $LP_{\mathcal{M}''} \geq (1-\epsilon)\cdot LP_{\mathcal{M}'}$. □

### 2.4.3 Characterizations of Policies in Step 5 of Algorithm 1

To estimate the performance of Step 6 in Algorithm 1, we need first to understand the nature of the online policies obtained in Step 5. The following lemma shows that they are optimal online policies subject to "shifting" values of elements.

**Lemma 6.** *Let $B \in \mathscr{S}$ be a maximal small bin. Then there exists $\lambda^*$ such that the online policy for $B$ computed in Step 5 is an optimal online policy for $B$ with values of $u_t$, $t \in B$ being $v_t - \lambda^*$.*

*Proof.* Let $\{\mathcal{X}_t^\star(\vec{s}, v_t), \mathcal{X}_t^\star(v_t), \mathcal{Y}_t^*(\vec{s}), \mathcal{N}_A^*\}$ indicate an optimal solution for ($LP$) as in Step 4. Note that the part of the optimal solution which corresponds to the elements from $B$ is also an optimal solution for the following linear program.

$$\begin{aligned} \text{maximize} \quad & \sum_{u_t \in B} \mathbb{E}_{v_t}[v_t \cdot \mathcal{X}_t(v_t)] \\ \text{subject to} \quad & \{\mathcal{X}_t(\vec{s}, v), \mathcal{X}_t(v), \mathcal{Y}_t(\vec{s})\} \in \mathcal{P}^B \\ & \sum_{u_t \in B} \mathbb{E}_{v_t}[\mathcal{X}_t(v_t)] = \mathcal{N}_B^\star \end{aligned}$$

Let us now consider the following relaxation of the above linear program by introducing $\lambda$.

$$\begin{aligned} \text{maximize} \quad & \sum_{u_t \in B} \mathbb{E}_{v_t}[v_t \cdot \mathcal{X}_t(v_t)] + \lambda(\mathcal{N}_B^* - \sum_{u_t \in B} \mathbb{E}_{v_t}[\mathcal{X}_t(v_t)]) \\ \text{subject to} \quad & \{\mathcal{X}_t(\vec{s}, v), \mathcal{X}_t(v), \mathcal{Y}_t(\vec{s})\} \in \mathcal{P}^B \end{aligned}$$

There exists $\lambda^*$ such that for $\lambda = \lambda^*$ every optimal solution for the first linear program is an optimal solution for the second linear program. Indeed, such a $\lambda^*$ can be obtained from the dual optimal solution for the first linear program.

Thus, the part of the optimal solution which corresponds to the elements from $B$ is also an optimal solution for the following linear program.

$$\begin{aligned} \text{maximize} \quad & \sum_{u_t \in B} \mathbb{E}_{v_t}[(v_t - \lambda^\star) \cdot \mathcal{X}_t(v_t)] \\ \text{subject to} \quad & \{\mathcal{X}_t(\vec{s}, v), \mathcal{X}_t(v), \mathcal{Y}_t(\vec{s})\} \in \mathcal{P}^B \end{aligned}$$

Note that the last linear program is encoding optimal online policies on the bin $B$, where all values for $v_t$ are shifted by $\lambda^*$. □

### 2.4.4 Properties of Optimal Online Policies on Left-to-Right LBOS

Now that we know that the online policies computed in Step 5 are optimal online policies for some left-to-right LBOS, let us understand the revenue and behavior of such optimal online policies. To study an optimal policy for a left-to-right LBOS instance $\widetilde{\mathcal{M}} = (\widetilde{U}, \widetilde{\mathcal{L}}, \widetilde{c})$, let us consider the natural dynamic program for finding such policies. For simplicity, each considered state consists of the current timestamp $t$ and the set $S \subseteq \widetilde{U}$ of currently selected elements. Let $D_t(S)$ indicate the expected revenue of the optimal pricing policy from items $\widetilde{u}_t$, $\widetilde{u}_{t+1}$, …, $\widetilde{u}_n$ given that we have currently selected items $S \subseteq \{\widetilde{u}_1, \ldots, \widetilde{u}_{t-1}\}$. These values of $D_t(S)$ can be defined using a dynamic programming approach as follows

$$D_t(S) := \begin{cases} D_{t+1}(S) & \text{if } S \cup \{\widetilde{u}_t\} \text{ is infeasible for } \widetilde{\mathcal{M}} \\ \mathbb{E}_{\widetilde{v}_t}\left[\max(D_{t+1}(S), D_{t+1}(S \cup \{\widetilde{u}_t\}) + \widetilde{v}_t)\right] & \text{otherwise} \end{cases}$$

together with

$$D_{n+1}(S) := 0\,.$$

Hence, in the case that an optimal online policy reached the state given by $S \subseteq \widetilde{U}$ and $t$, then this optimal policy selects an element $\widetilde{u}_t$ with value $\widetilde{v}_t$ if and only if $S \cup \{\widetilde{u}_t\}$ is feasible and

$$\widetilde{v}_t \geq D_{t+1}(S) - D_{t+1}(S \cup \{\widetilde{u}_t\})\,.$$

In other words, $D_{t+1}(S) - D_{t+1}(S \cup \{\widetilde{u}_t\})$ is the threshold value at which the optimal online policy selects $\widetilde{u}_t$.

The following lemma shows that slight changes in the current state set $S$ lead only to slight changes in the number of elements selected by the optimal policy in the future.

**Lemma 7.** *Let $\widetilde{\mathcal{M}} = (\widetilde{U}, \widetilde{\mathcal{L}}, \widetilde{c})$ be a left-to-right LBOS instance with ordering of elements $\widetilde{u}_1$, $\widetilde{u}_2$, …, $\widetilde{u}_n$. Consider the selections done by an optimal online policy when the policy encounters a fixed realization of values $\widetilde{v}_2$, $\widetilde{v}_3$, …, $\widetilde{v}_n$. For this fixed realization, let $\mu_0$ and $\mu_1$ be the number of elements selected by an optimal online policy from $\{\widetilde{u}_2, \ldots, \widetilde{u}_n\}$ starting at $t = 2$ with $S$ being $\varnothing$ and $\{\widetilde{u}_1\}$, respectively. Then we have that*

$$\mu_1 \leq \mu_0 \leq \mu_1 + 1\,.$$

Note that Lemma 7 is essentially the same as saying that the function returning the number of items selected by the optimal solution, given the realizations of values, is 1-Lipschitz. Thus, there is an alternative path to showing the needed concentration results that goes through Talagrand's inequality (see, for example, [MR02]).

To show Lemma 7, we first need the following lemma.

**Lemma 8.** *Let $\widetilde{\mathcal{M}} = (\widetilde{U}, \widetilde{\mathcal{L}}, \widetilde{c})$ be a left-to-right* LBOS *instance with ordering of elements $\widetilde{u}_1, \ldots, \widetilde{u}_n$. Let $S$, $I$, $I' \subseteq \{\widetilde{u}_1, \ldots, \widetilde{u}_{t-1}\}$ be pairwise disjoint sets. Let $I \cup I'$ be the set $\{i_1, i_2, \ldots, i_k\}$, where $i_1$, $i_2$, $\ldots$, $i_k$ is the arrival order of elements from $I \cup I'$. Let us define two subsets of $I \cup I'$ using the parity of their indices*

$$I_{odd} := \{i_1, i_3, \ldots\} \qquad \text{and} \qquad I_{even} := \{i_2, i_4, \ldots\}\,.$$

*If $S \cup I, S \cup I', S \cup I_{odd}$, and $S \cup I_{even}$ are feasible with respect to $\widetilde{\mathcal{M}}$ then we have*

$$D_t(S \cup I) + D_t(S \cup I') \leq D_t(S \cup I_{odd}) + D_t(S \cup I_{even})\,, \tag{2.6}$$

*for all $t$.*

*Proof.* We show (2.6) by induction on $t$. When $t = n + 1$, the statement is true as $D_{n+1}(S) = 0$ for all $S \subseteq U$. Suppose that (2.6) holds for $t + 1$.

It is crucial for us to understand the thresholds that the value $v_t$ of $u_t$ should "pass" in order to be selected given that the current state is $S'$. These thresholds have the following structure.

$$T_t(S') := \begin{cases} D_{t+1}(S') - D_{t+1}(S' \cup \{u_t\}) & \text{if } S' \cup \{u_t\} \text{ is feasible for } \widetilde{M} \\ +\infty & \text{otherwise}\,. \end{cases}$$

Let us fix the value of $\widetilde{v}_t$ and define the following values for $S' \subseteq \{\widetilde{u}_1, \ldots, \widetilde{u}_{t-1}\}$.

$$F_t(S') := \begin{cases} D_{t+1}(S') & \text{if } S' \cup \{\widetilde{u}_t\} \text{ is infeasible for } \widetilde{\mathcal{M}} \\ \max(D_{t+1}(S'), D_{t+1}(S' \cup \{\widetilde{u}_t\}) + \widetilde{v}_t) & \text{otherwise} \end{cases}$$

Instead of showing (2.6), we show that the following inequality holds

$$F_t(S \cup I) + F_t(S \cup I') \leq F_t(S \cup I_{odd}) + F_t(S \cup I_{even})\,. \tag{2.7}$$

By linearity and by the definition of $D_t(\cdot)$, the inequality (2.7) implies the inequality (2.6). We consider three cases based on the behavior of $\widetilde{v}_t$ with respect to the thresholds $T_t(S \cup I)$ and $T_t(S \cup I')$.

**Case $\widetilde{v}_t < T_t(S \cup I)$ and $\widetilde{v}_t < T_t(S \cup I')$.** Then $\widetilde{u}_t$ is selected at neither of these states, so we have that

$$F_t(S \cup I) + F_t(S \cup I') = D_{t+1}(S \cup I) + D_{t+1}(S \cup I')\,,$$

and by the definition of $D_t(\cdot)$ and $F_t(\cdot)$ we have

$$F_t(S \cup I_{odd}) + F_t(S \cup I_{even}) \geq D_{t+1}(S \cup I_{odd}) + D_{t+1}(S \cup I_{even}),$$

proving that the inductive step is valid.

**Case $\widetilde{v}_t \geq T_t(S \cup I)$ and $\widetilde{v}_t \geq T_t(S \cup I')$.** Then $\widetilde{u}_t$ is accepted at both of these states, so we have that

$$F_t(S \cup I) + F_t(S \cup I') = D_{t+1}(S \cup I \cup \{\widetilde{u}_t\}) + D_{t+1}(S \cup I' \cup \{\widetilde{u}_t\}) + 2\widetilde{v}_t,$$

Note that for every bin $B \in \widetilde{\mathcal{L}}$ due to the left-to-right arrival order and the definition of $I_{odd}$ and $I_{even}$, we have that the elements from $B$ in $I_{odd} \cup I_{even}$ are evenly distributed:

$$|B \cap I_{even}| - 1 \leq |B \cap I_{odd}| \leq |B \cap I_{even}| + 1.$$

This implies that

$$\max\{|B \cap I_{odd}|, |B \cap I_{even}|\} \leq \max\{|B \cap I|, |B \cap I'|\}.$$

Thus, since both sets $S \cup I \cup \{\widetilde{u}_t\}$ and $S \cup I' \cup \{\widetilde{u}_t\}$ are feasible with respect to $\widetilde{M}$, the sets $S \cup I_{odd} \cup \{\widetilde{u}_t\}$ and $S \cup I_{even} \cup \{\widetilde{u}_t\}$ are feasible with respect to $\widetilde{M}$. So by the inductive hypothesis and definition of $D_t(\cdot)$ and $F_t(\cdot)$, we have

$$\begin{aligned} F_t(S \cup I_{odd}) + F_t(S \cup I_{even}) &\geq D_{t+1}(S \cup \{\widetilde{u}_t\} \cup I_{odd}) + D_{t+1}(S \cup \{\widetilde{u}_t\} \cup I_{even}) + 2\widetilde{v}_t \\ &\geq D_{t+1}(S \cup \{\widetilde{u}_t\} \cup I) + D_{t+1}(S \cup \{\widetilde{u}_t\} \cup I') + 2\widetilde{v}_t \\ &= F_t(S \cup I) + F_t(S \cup I'). \end{aligned}$$

**Case when exactly one of $\widetilde{v}_t \geq T_t(S \cup I)$ or $\widetilde{v}_t \geq T_t(S \cup I')$ holds.** Suppose without loss of generality that $\widetilde{v}_t \geq T_t(S \cup I)$ holds. We have two cases depending on the parity of $k$ in the statement of the lemma. Let us suppose further that $k$ is even; the case when $k$ is odd is completely analogous. Analogously to the above cases, both $S \cup I_{odd} \cup \{\widetilde{u}_t\}$ and $S \cup I_{even}$ are feasible with respect to $\widetilde{\mathcal{M}}$ and we obtain

$$F_t(S \cup I_{odd}) + F_t(S \cup I_{even}) \geq F_t(S \cup I) + F_t(S \cup I').$$

□

*Proof of Lemma 7.* Let $J$ and $J'$ be the elements selected by the optimal online policy starting at $t = 2$ with $S = \varnothing$ and $S = \{\widetilde{u}_1\}$, respectively. Let $S := J \cap J'$, $I := J \setminus J'$ and $I' := J' \setminus J$ and let $I_{odd}$, $I_{even}$ be defined as in Lemma 7.

To prove the statement of the lemma, we show that the following properties hold

$$I'_t := I' \cap \{\widetilde{u}_1, \dots, \widetilde{u}_t\} = I_{odd} \cap \{\widetilde{u}_1, \dots, \widetilde{u}_t\} \qquad \text{(ALT)}$$
$$I_t := I \cap \{\widetilde{u}_1, \dots, \widetilde{u}_t\} = I_{even} \cap \{\widetilde{u}_1, \dots, \widetilde{u}_t\}$$

for all $t = 1, \dots, n$. The property (ALT) shows that the behavior of the optimal online policies "alternates" starting from $S = \varnothing$ and $S = \{\widetilde{u}_1\}$ after the first timestamp. Thus, (ALT) immediately implies the statement of the lemma.

Let us prove the statement by induction on $t$. For $t = 1$ the statement is trivial. Let us assume that (ALT) holds for the timestamp $t-1$ and let us prove that it holds also for $t$. We have two possible cases: $|I'_{t-1}| = |I_{t-1}| + 1$ and $|I'_{t-1}| = |I_{t-1}|$. Let us consider only the case $|I'_{t-1}| = |I_{t-1}| + 1$ as the other case is completely analogous.

Let $S_{t-1}$ be defined as $S \cap \{\widetilde{u}_1, \dots, \widetilde{u}_{t-1}\}$. Consider the sets $S_{t-1} \cup I_{t-1} \cup \{\widetilde{u}_t\}$ and $S_{t-1} \cup I'_{t-1} \cup \{\widetilde{u}_t\}$. Note that with respect to $\widetilde{M}$ they are either both infeasible, only the latter is infeasible, or they are both feasible. For the inductive step, the only non-trivial case is when they are both feasible with respect to $\widetilde{M}$. The threshold for selecting $\widetilde{u}_t$ in the case that the current state corresponds to $S_{t-1} \cup I_{t-1}$ is

$$D_{t+1}(S_{t-1} \cup I_{t-1}) - D_{t+1}(S_{t-1} \cup I_{t-1} \cup \{\widetilde{u}_t\})$$

and in the case that the current state corresponds to $S_{t-1} \cup I'_{t-1}$, the threshold is

$$D_{t+1}(S_{t-1} \cup I'_{t-1}) - D_{t+1}(S_{t-1} \cup I'_{t-1} \cup \{\widetilde{u}_t\})\,.$$

By Lemma 7 and by (ALT) for the timestamp $t-1$, we have

$$D_{t+1}(S_{t-1} \cup I_{t-1} \cup \{\widetilde{u}_t\}) + D_{t+1}(S_{t-1} \cup I'_{t-1}) \geq D_{t+1}(S_{t-1} \cup I'_{t-1} \cup \{\widetilde{u}_t\}) + D_{t+1}(S_{t-1} \cup I_{t-1})\,.$$

Thus the threshold for selecting $\widetilde{u}_t$, when the current state corresponds to $S_{t-1} \cup I_{t-1}$, is not larger than the same threshold when the current state corresponds to $S_{t-1} \cup I'_{t-1}$. Thus, $\widetilde{u}_t$ has passed both thresholds or only the threshold corresponding to $S_{t-1} \cup I_{t-1}$, showing (ALT) for the timestamp $t$. In both cases, (ALT) holds for the timestamp $t$. □

### 2.4.5 Results to Prove Concentration

Now that we have characterized the actions of our policy on small bins, we can return to the proof of Lemma 3. The proof of Lemma 3 heavily relies on Lemma 7 and is the key ingredient we used in the proof of Lemma 4. In particular, Lemma 3 allowed us to obtain several estimates for the number of elements suggested to be selected from $B$ in Step 5 as if the suggestions for elements were negatively correlated.

*Proof of Lemma 3.* In the proof, we use only the fact that the online policies computed in Step 5 are optimal online policies for $B$. Thus, these online policies have to satisfy Lemma 7. We show the claim by induction on the total number $k$ of elements in the bin $B$. The statement trivially holds when $k = 0$. Suppose that the statement is true for all instances with $k-1$ elements. Abusing the notation, let $u_1$ be the first element in $B$ to arrive. Define $\mu := \mathbb{E}[\sum_{u_t \in B} X_t]$, $\mu_1 := \mathbb{E}[\sum_{u_t \in B,\, t \neq 1} X_t \,|\, X_1 = 1]$, $\mu_0 := \mathbb{E}[\sum_{u_t \in B,\, t \neq 1} X_t \,|\, X_1 = 0]$ and $p_1 = \mathbf{Pr}[X_1 = 1]$, $p_0 = \mathbf{Pr}[X_1 = 0]$.

By the inductive hypothesis, we have

$$\begin{aligned}\mathbb{E}[e^{\alpha X_B}] &= p_1 \cdot e^{\alpha} \cdot \mathbb{E}[e^{\sum_{i=2}^{n} \alpha X_i} \,|\, X_1 = 1] + p_0 \cdot \mathbb{E}[e^{\sum_{i=2}^{n} \alpha X_i} \,|\, X_1 = 0] \\ &\leq p_1 \cdot e^{\alpha} \cdot e^{(e^{\alpha}-1)\cdot\mu_1} + p_0 \cdot e^{(e^{\alpha}-1)\cdot\mu_0}\,.\end{aligned}$$

Consider the following program where $\mu$ is a constant, while $p_0$, $p_1$, $\mu_0$, and $\mu_1$ are considered variables.

$$\begin{aligned}&\text{maximize} && p_1 \cdot e^{\alpha} \cdot e^{(e^{\alpha}-1)\cdot\mu_1} + p_0 \cdot e^{(e^{\alpha}-1)\cdot\mu_0} \\ &\text{subject to} && p_0 + p_1 = 1, \qquad 0 \leq p_0 \leq 1 \\ & && p_1 \cdot \mu_1 + p_1 + p_0 \cdot \mu_0 = \mu, \qquad \mu_1 \leq \mu_0 \leq \mu_1 + 1\end{aligned}$$

Note that the last inequality holds due to Lemma 7. To prove the current lemma, we show that the objective value of this program is bounded from above by $e^{(e^{\alpha}-1)\cdot\mu}$. Let $p_0^*, p_1^*, \mu_0^*, \mu_1^*$ be a feasible solution of this program.

**Case $p_1^* = 1$.** Then we must have that $\mu = \mu_1^* + 1$, and so the objective value of such a solution becomes $e^{\alpha} \cdot e^{(e^{\alpha}-1)\cdot(\mu-1)} \leq e^{(e^{\alpha}-1)\cdot\mu}$, where the inequality follows from $\alpha \leq e^{\alpha-1}$.

**Case $p_1^* = 0$.** Then the objective value of such a solution is exactly $e^{(e^{\alpha}-1)\cdot\mu}$.

**Case $p_1^* \in (0,1)$.** We obtain the following equation $\mu_1^* = (\mu - \mu_0^* p_0^* - p_1^*)/p_1^*$. From now on we can consider $p_0^*$, $p_1^*$ as constants, and so the program now corresponds to an optimization over a single variable $\mu_0^*$, and further, the objective function is convex in this variable. Therefore, it suffices to check that $\mu_0^*$ on the boundary of the arising feasibility region corresponds to an objective value that is at most $e^{(e^{\alpha}-1)\cdot\mu}$.

If $\mu_1^* + 1 = \mu_0^*$, then $\mu_0^* = \mu$, and the objective value is

$$p_1^* \cdot e^{\alpha} \cdot e^{(e^{\alpha}-1)\cdot(\mu-1)} + p_0^* \cdot e^{(e^{\alpha}-1)\cdot\mu} \leq p_1^* \cdot e^{(e^{\alpha}-1)\cdot\mu} + p_0^* \cdot e^{(e^{\alpha}-1)\cdot\mu} = e^{(e^{\alpha}-1)\cdot\mu},$$

where as before, we use the inequality $\alpha \leq e^{\alpha-1}$.

If $\mu_1^* = \mu_0^*$, then $\mu_1^* + p_1^* = \mu$, and the objective value is

$$e^{(e^{\alpha}-1)\cdot(\mu-p_1^*)} \cdot (p_1^* \cdot e^{\alpha} + p_0^*) \leq e^{(e^{\alpha}-1)\cdot\mu},$$

using the fact that $p_1^* \cdot e^\alpha + p_0^* = p_1^* \cdot e^\alpha + (1 - p_1^*) \leq e^{(e^\alpha - 1) \cdot p_1}$.

□

### 2.4.6 Putting Everything Together

**Theorem 2.** *Algorithm 1 is a PTAS for left-to-right* `LBOS` *instances when* $K = \epsilon^{-4}$.

*Proof.* First, note that the set of elements selected by Algorithm 1 is independent with respect to $\mathcal{M}'$, and so by Lemma 2 is independent with respect to $\mathcal{M}$.

The online policies that are extracted in Step 5 of Algorithm 1 correspond to running online policies independently on maximal small bins. When run on maximal small bins independently, these online policies guarantee the revenue $LP_{\mathcal{M}''}$, which is at least $(1 - \epsilon)^2 \cdot OPT_{\mathcal{M}}$ by Lemma 5.

However, in Step 6 the online policies from Step 5 can fail to select some elements only because of the feasibility constraints. By Lemma 4 and since $K = \epsilon^{-4}$, for each $u_t \in U$ the probability that upon the arrival of $u_t$ the online policies suggest selecting $u_t$ but it is not feasible is at most $3\epsilon$. Thus, the expected revenue obtained by Algorithm 1 is at least $(1-3\epsilon)LP_{\mathcal{M}''}$, and so at least $(1-\epsilon)^2(1-3\epsilon) \cdot OPT_{\mathcal{M}} \geq (1-5\epsilon) \cdot OPT_{\mathcal{M}}$. Re-parameterizing $\epsilon$ by a constant factor completes the proof.

□

## 2.5 Extending to Arrivals "Close" to Left-to-Right

In this section, we extend the results for left-to-right `LBOS` instances to `LBOS` instances that are "close" to being left-to-right. Let $\mathcal{M} = (U, \mathcal{L}, c)$ be an `LBOS` instance and $B \in \mathcal{L}$ a bin. We say that $B$ is a *left-to-right bin* if the instance $\mathcal{M}$ restricted to $B$ is a left-to-right `LBOS` instance.

We are able to extend the results because we do not need every bin in laminar matroid to be left-to-right. For our analysis to go through, it is enough for every maximal small bin to be left-to-right. Given the above observation, we obtain the following theorem by using a modified definition of small and big bins, which is inspired by [ANSS19].

**Theorem 3.** *Let $L \in \mathbb{N}$ be a constant. There is a PTAS for* `LBOS` *instances where every element lies in at most $L$ bins that are not left-to-right.*

*Proof Sketch.* In [ANSS19], a PTAS is given for constant-depth LBOS instances. For their algorithm, they have a different definition of "small" and "big" bins. In Algorithm 1 we use one threshold $K = \epsilon^{-4}$ for all bins: a bin $A$ is big for our Algorithm 1 if $c(A) \geq K$. In [ANSS19], there are several thresholds: a bin $A$ is "big" for them if $c(A) \geq 1/\delta^{L-d}$, where $\delta = \epsilon^2/\log(1/\epsilon)$, $d$ is the depth of $A$, and $L$ is the depth of the whole matroid. The benefit of the approach from [ANSS19] is that for any "big" bin $B$ and maximal "small" bin $A \subseteq B$, we have a separation between $c(A)$ and $c(B)$, in particular, we have $c(A) \leq \delta \cdot c(B)$. This large separation in capacities removes the need to analyze the concentration for the number of elements selected from $A$. In particular, there is no need for $A$ to be a left-to-right bin. Constant depth is needed to bound the capacity of any maximal "small" bin by a constant $1/\delta^L$ in order to explicitly solve the corresponding dynamic program.

To generalize this to our setting, we use the following thresholds to define "big" and "small" bins. If the depth $d$ of a bin $A$ satisfies $d \leq L$, then we set the threshold for $A$ to be $\epsilon^{-4}/\delta^{L-d}$. Otherwise, we set the threshold for $A$ to be $\epsilon^{-4}$. If a maximal "small" bin is identified before depth $L$, then it will necessarily have the factor of $\delta$ separation with any "big" bins containing it. Otherwise, we know that the identified maximal "small" bin is a left-to-right bin. □

**Corollary 2** (Theorem 3.5 from [ANSS19])**.** *There is a PTAS for constant-depth* ***LBOS*** *instances.*

The production constrained Bayesian selection problem from [ANSS19] consists of LBOS instances in which all bins except the largest are left-to-right, plus some additional structure. This gives us the following corollary with $L = 1$.

**Corollary 3** (Theorem 2.3 from [ANSS19])**.** *There is a PTAS for production constrained Bayesian selection.*

## 2.6 Laminar Instances with Bad Concentration

Our PTAS for left-to-right Laminar Bayesian Online Selection relies crucially on Lemma 3, which show that the number of elements selected by an optimal solution is well-concentrated around the mean. The only point in our analysis in which we needed the left-to-right arrival order was in the proof of this Lemma. It is natural to wonder how far such a result can be pushed - in particular, how bad can the concentration for general laminar instances be. In this section, we show that there are laminar instances in which the optimal solution demonstrates essentially worst-possible concentration on the number of selected elements.

**Theorem 4.** *For any $r \geq 1$ and $\epsilon > 0$, there exists an instance of laminar Bayesian online selection $\mathcal{M} = (U, \mathcal{L}, c)$ with rank$(\mathcal{M}) = r$ and $|U| \leq r^2$ such that, letting $OPT(\mathcal{M}) \subseteq U$ be a random variable denoting the items selected by the optimal algorithm,*

$$\boldsymbol{Pr}[|OPT(\mathcal{M})| = 0], \boldsymbol{Pr}[|OPT(\mathcal{M})| = r] \in [\frac{1}{2} - \epsilon, \frac{1}{2} + \epsilon].$$

We will first describe the intuition behind our construction. Consider the instance described in Figure 2.1.

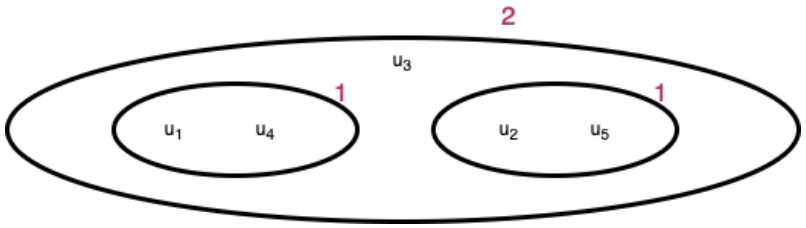



Figure 2.1: The elements arrive in order $u_1, u_2, \ldots, u_5$ with values $v_1, v_2, \ldots, v_5$. $v_1$ is uniformly distributed on $\{0, 10\}$. $v_2$ is equal to 1.1. $v_3$ is uniformly distributed on $\{0, 1\}$. $v_4$ and $v_5$ are uniformly distributed on $\{0, 2\}$. This instance was originally described in [ANSS19].

It is clear to see that $u_1$ will be selected whenever it has value 10. If $u_1$ is selected, then only one of $u_2, u_3, u_5$ may be selected in the future. The optimal strategy is then to select $u_2$. Suppose that $u_1$ is not selected. If $u_2$ is also not selected, then any two of $u_3, u_4, u_5$ may be selected in the future, yielding expected value 2.25. On the other hand, if $u_2$ is selected, then only one of $u_3$ and $u_4$ may be selected in the future, with an expected value of 1. Even after adding in the gained value for $u_2$, this amounts to only 2.1, so it is optimal to not select $u_2$ when $u_1$ is not selected.

So we have seen that in the instance from Figure 2.1, the selection of $u_1$ and $u_2$ are perfectly correlated - half the time $\{u_1, u_2\}$ is selected, and half the time, neither are selected. That is, when $u_1$ is not selected, $u_2$ should be discarded to preserve room for the remaining elements. But when $u_1$ is selected, we can select $u_2$ without worry as there are less candidate elements in the future to save room for. By carefully increasing the values while decreasing probability for each of $u_3, u_4$, and $u_5$, this behavior can be maintained while making the probability of $u_3, u_4$, or $u_5$ having non-zero value arbitrarily low. So with the right choice of distributions, we can create an anti-concentrated instance for $r = 2$.

To generalize to $r = 3$, we add an extra "layer" to our instance, see Figure 2.2.

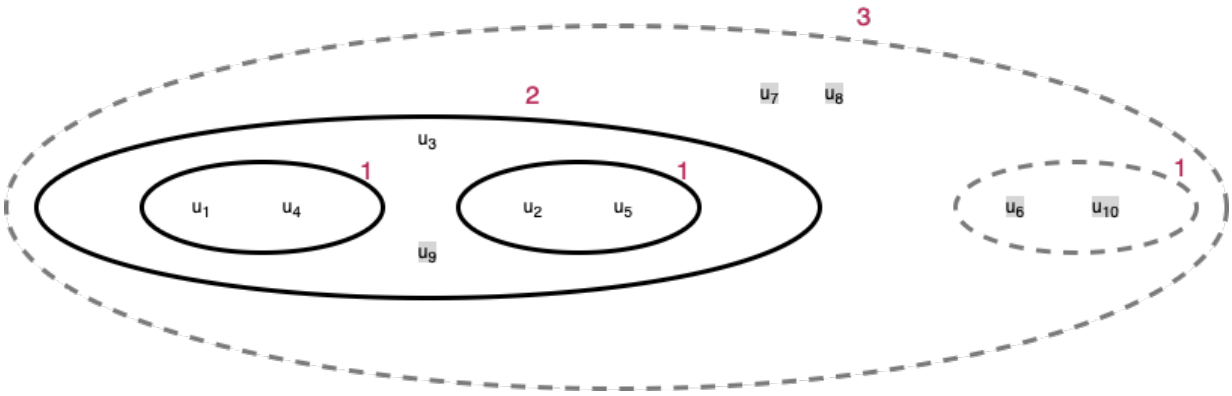



Figure 2.2: The laminar family, capacities, and arrival order generated by Algorithm 2 when $r = 3$. New sets in the laminar family are drawn with dashed lines, and new items are highlighted.

Here, we forego giving exact values for $u_6$ through $u_{10}$. Importantly, these values set to be are incredibly small compared to the values of $u_1$ through $u_5$. Because they are so small, these new elements will have no effect on the behavior of the optimal algorithm on $u_1$ through $u_5$. So as observed above, the optimal algorithm will almost always select $\{u_1, u_2\}$ or nothing on the first 5 elements. Similarly to the rank 2 case, we can carefully set the distributions for $u_6, u_7, \ldots, u_{10}$ so that $u_6$ is selected precisely when $\{u_1, u_2\}$ were already selected. This general method of expansion can be continued for higher ranks.

*Proof of Theorem 4.* We now present Algorithm 2, which provides anti-concentrated instances for all ranks. For simplicity, we will label the elements of the matroid as $U = \{u_1, u_2, \ldots, u_n\}$. For each element $u_i$, the produced instance uses only weighted Bernoulli distributions with support $\{0, v_i\}$ for some $v_i > 0$. Let $p_i$ be the probability that $u_i$ takes on value $v_i$. Thus, with probability $1 - p_i$, the element $u_i$ takes on value 0.

Given natural numbers $r$ and $k$, Algorithm 2 generates an anti-concentrated instance $\mathcal{M}$. Here, the choice of $k$ is later determined by the value of $\varepsilon$.

Note that the total number of items is $|U_r| = 1 + \sum_{i=2}^{r} i + 2 \leq (r+1)^2$.

The analysis proceeds by showing that the optimal online algorithm will select $u_1$ with probability $\frac{1}{2}$, in which case it will then also select $u_{n_1+1}, u_{n_2+1}, \ldots, u_{n_{r-1}+1}$. And when the optimal algorithm does not select $u_1$, it will almost surely select no elements at all.

**Algorithm 2:** PRODUCE-INSTANCE(r, k)

$v \leftarrow 10^{1+2k}, \quad v' \leftarrow 10^k, \quad p \leftarrow 10^{-1-k}$ ;
$n_1 \leftarrow 1, \quad U_1 \leftarrow \{u_1\}$ ;
$v_1 \leftarrow 1, \quad p_1 \leftarrow \frac{1}{2}$ ;
$\Delta_1 \leftarrow 1$ ;
$\mathcal{L}_1 \leftarrow \{\{u_1\}\}, \quad c_1(\{u_1\}) \leftarrow 1$ ;
**for** $i = 2, 3, \ldots, r$ **do**
  $\alpha_i \leftarrow \frac{\Delta_{i-1}}{10(r+1)^2} \cdot \frac{1}{v}$ ;
  $U_i \leftarrow U_{i-1} \cup \{u_{n_{i-1}+1}, \ldots, u_{n_{i-1}+i+2}\}, \quad n_i \leftarrow n_{i-1} + i + 2$ ;
  $\mathcal{L}_i \leftarrow (\mathcal{L}_{i-1} \setminus \{U_{i-1}\}) \cup \{U_{i-1} \cup \{u_{n_{i-1}+i+1}\}, U_i, \{u_{n_{i-1}+1}, u_{n_{i-1}+i+2}\}\}$ ;
  $c_i(B) \leftarrow c_{i-1}(B)$ for all $B \in \mathcal{L}_{i-1} \cap \mathcal{L}_i$ ;
  $c_i(U_{i-1} \cup \{u_{n_{i-1}+i+1}\}) \leftarrow i - 1, \quad c_i(U_i) \leftarrow i, \quad c_i(\{u_{n_{i-1}+1}, u_{n_{i-1}+i+2}\}) \leftarrow 1$ ;
  $\Delta_i \leftarrow \alpha_i \cdot v \cdot p^i \cdot (1-p)$ ;
  $p_{n_{i-1}+1} \leftarrow 1, \quad v_{n_{i-1}+1} \leftarrow \alpha_i \cdot 10^k + \frac{\Delta_i}{2}$ ;
  $p_{n_{i-1}+2}, \ldots, p_{n_{i-1}+i+2} \leftarrow p$ ;
  $v_{n_{i-1}+2}, \ldots, v_{n_{i-1}+i} \leftarrow \alpha_i \cdot v'$ ;
  $v_{n_{i-1}+i+1}, \quad v_{n_{i-1}+i+2} \leftarrow \alpha_i \cdot v$ ;
**end**
return $\mathcal{M} = (U_r, \mathcal{L}_r, c_r)$ with values $v_1, \ldots, v_{n_r}$ and probabilities $p_1, \ldots, p_{n_r}$ ;

**Lemma 9.** *For every $i \geq 2$, the probability that we have $OPT(\mathcal{M}) \cap \{u_1 \ldots, u_{n_i}\} = \varnothing$, conditioned on $OPT(\mathcal{M}) \cap \{u_1 \ldots, u_{n_{i-1}}\} = \varnothing$, is at least $(1-p)^{i+1}$.*

*Proof.* Re-label the elements so that $(w_1, w_2, \ldots, w_{i+2}) = (u_{n_{i-1}+1}, u_{n_{i-1}+2}, \ldots, u_{n_{i-1}+i+2})$. Let us show that, conditioned on $OPT(\mathcal{M}) \cap \{u_1 \ldots, u_{n_{i-1}}\} = \varnothing$, the optimal policy does not select $w_1$. This suffices to show the statement of the lemma, since the probability that all elements among $w_2, \ldots, w_{i+2}$ have value zero, and hence will not be selected by the optimal algorithm, is $(1-p)^{i+1}$.

Now let us condition on $OPT(\mathcal{M}) \cap \{u_1 \ldots, u_{n_{i-1}}\} = \varnothing$ and estimate the expected revenue of strategies that select and do not select $w_1$.

First, consider a strategy that does not select $w_1$ and after that greedily accepts every element among $w_2, \ldots, w_{i+2}$ that has non-zero value and is feasible. The only situation in which this strategy will reject an element with a non-zero value is in the case that all of $w_2, \ldots, w_{i+2}$ have non-zero values, in which case it will reject $w_{i+2}$. So from each of these elements, we gain their expected value, with a small loss for $w_{i+2}$. On the elements $w_2$,

$\dots w_{i+2}$, this strategy yields expected value

$$\alpha_i \cdot \left((i-1) \cdot v' \cdot p + 2 \cdot v \cdot p - v \cdot p^{i+1}\right) .$$

Now consider any strategy that selects the element $w_1$. Clearly, such a strategy cannot select $w_{i+2}$ due to the constraints in the laminar matroid. To maximize expected revenue, an optimal strategy must greedily select all items from $w_2, \dots, w_{i+1}$ as long as they have non-zero value and are feasible. As above, this strategy will always take any item in $w_2, \dots, w_{i+1}$ with non-zero value, except in the case that all these elements take on non-zero value. Therefore, any strategy on elements $w_2, \dots, w_{i+2}$ will yield expected value at most

$$\alpha_i \cdot \left((i-1) \cdot v' \cdot p + v \cdot p - v \cdot p^{i}\right) .$$

Note that so far we considered only revenues on the elements $w_2, \dots w_{i+2}$. However, there are elements with arrivals after the arrival of $w_{i+2}$. Due to our scaling with $\alpha_{i+1}, \alpha_{i+2}, \dots$, we know that the value of any element after $w_{i+2}$ is at most $\Delta_i/(10(r+1)^2)$. There are at most $(r+1)^2$ elements arriving after $w_{i+2}$, so the sum of their value is at most $\Delta_i/10$. Therefore, if a strategy selects $w_1$, its expected value from elements arriving after $w_1$ is at most

$$\alpha_i \cdot \left((i-1) \cdot v' \cdot p + v \cdot p - v \cdot p^{i}\right) + \frac{\Delta_i}{10}.$$

So for the optimal strategy to select $w_1$, the value of $w_1$ should be at least

$$\begin{aligned}
&\alpha_i \cdot \left((i-1) \cdot v' \cdot p + 2 \cdot v \cdot p - v \cdot p^{i+1}\right) - \alpha_i \cdot \left((i-1) \cdot v' \cdot p + v \cdot p - v \cdot p^{i}\right) + \frac{\Delta_i}{10} \\
=&\alpha_i \cdot \left(v \cdot p - v \cdot p^{i+1} + v \cdot p^{i}\right) - \frac{\Delta_i}{10} \\
=&\alpha_i \cdot 10^k + \Delta_i - \frac{\Delta_i}{10} \\
>&\alpha_i \cdot 10^k + \frac{\Delta_i}{2}.
\end{aligned}$$

Therefore, the optimal solution does not select $w_1$ as it has value $\alpha_i \cdot 10^k + \Delta_i/2$. □

**Lemma 10.** *For every $i \geq 2$, the probability that we have $|OPT(\mathcal{M}) \cap \{u_1 \dots, u_{n_i}\}| = i$, conditioned on $|OPT(\mathcal{M}) \cap \{u_1 \dots, u_{n_{i-1}}\}| = i-1$, equals* 1.

*Proof.* Re-label the elements so that $(w_1, w_2, \ldots, w_{i+2}) = (u_{n_{i-1}+1}, u_{n_{i-1}+2}, \ldots, u_{n_{i-1}+i+2})$. We will show that, conditioned on $|OPT(\mathcal{M}) \cap \{u_1 \ldots, u_{n_{i-1}}\}| = i - 1$, the optimal policy selects $w_1$. To do so, we analyze the performance of possible strategies on all elements after $w_{i+2}$ in both the situation that $w_1$ is selected and the situation that it is skipped.

If $w_1$ is skipped by a strategy, then to obtain optimal revenue on $w_2, \ldots, w_{i+2}$, the strategy must greedily select the first item that has non-zero value, noting that $w_{i+1}$ cannot be selected as it is contained in the saturated constraint corresponding to $U_{i-1} \cup \{w_{i+1}\}$. The probability that at least one of $w_2$, …, $w_i$ has a non-zero value is $(1 - (1-p)^{i-1})$, in which case we gain value $\alpha_i \cdot v'$. Otherwise, with probability $(1-p)^{i-1} \cdot p$, none of $w_2, \ldots, w_i$ have a non-zero value but $w_{i+2}$ does, and so a strategy gains the value $\alpha_i \cdot v$. Hence, a strategy skipping the element $w_1$ will gain in expectation on the elements $w_2 \ldots, w_{i+2}$ at most

$$\alpha_i \cdot \left(v' \cdot \left(1 - (1-p)^{i-1}\right) + v \cdot p \cdot (1-p)^{i-1}\right) = \alpha_i \cdot 10^k.$$

As seen in the proof of Lemma 9, the total value of all elements after $w_{i+2}$ is bounded by $\Delta_i/10$. So the expected revenue of a strategy skipping $w_1$ is at most $\alpha_i \cdot 10^k + \Delta_i/10$.

Now, note that $\alpha_i \cdot 10^k + \Delta_i/10 < \alpha_i \cdot 10^k + \Delta_i/2$, the value of $w_1$. Hence, the optimal strategy selects $w_1$, as wanted.

□

Finally, we note that item $u_1$ is taken by the optimal algorithm whenever its value is non-zero. So the optimal strategy selects $u_1$ with probability 1/2. Applying Lemmas 9 and 10 inductively, this implies that

$$\mathbf{Pr}[\,|OPT(\mathcal{M})| = 0\,] \geq \frac{1}{2} \cdot (1-p)^{(r+1)^2}$$

and

$$\mathbf{Pr}[\,|OPT(\mathcal{M})| = r\,] \geq \frac{1}{2}.$$

Selecting sufficiently large $k$, we have $(1-p)^{(r+1)^2} \geq (1-\epsilon)$, obtaining the statement of Theorem 4.

□

# Chapter 3

# Hardness of Matroid Bayesian Online Selection

LBOS can be viewed as a special case of the more general Matroid Bayesian Online Selection problem, where instead of a laminar matroid, we are given any matroid $\mathcal{M}$. Gupta questioned in 2017 whether the best strategy for Matroid Bayesian Online Selection is computationally hard to find [Gup17]. We answer this question in the affirmative, even for the special case of graphic matroids.

**Theorem 5.** *There is an absolute constant $\alpha \in (0, 1)$ such that it is PSPACE-hard to approximate Graphic Bayesian Online Selection (GBOS) to a factor of $\alpha$.*

Note that we can compute the optimal strategy for GBOS in polynomial space with a simple brute-force recursive algorithm, so GBOS is in fact PSPACE-complete.

Also note that Theorem 5 says it is PSPACE-hard to approximate the *expected value* of an optimal strategy. This does not immediately imply that it is PSPACE-hard to act according to an approximate optimal strategy, but the proof of Theorem 5 shows this as well.

**Definition 2.** *We say that a class of matroids $\mathscr{M}$* admits a PTAS-compatible distribution-agnostic order *if there exists an order $\sigma(\mathcal{M})$ for every matroid $\mathcal{M} \in \mathscr{M}$ such that there is a Polynomial-Time Approximation Scheme for instances of Matroid Bayesian Online Selection with matroids from $\mathscr{M}$ and arrival orders given by $\sigma$.*

We show the following result by demonstrating that the hardness construction of Theorem 5 can be embedded in any arrival order of sufficiently large complete graphs.

**Theorem 6.** *The set of graphic matroids does not admit a PTAS-compatible distribution-agnostic order unless* PSPACE = P.

As a corollary of Theorem 2, we have that the set of laminar matroids admits a PTAS-compatible distribution-agnostic order by taking $\sigma(\mathcal{M})$ to be any left-to-right ordering on $\mathcal{M}$. Theorem 6 provides a separation between graphic and laminar matroids.

We also show hardness for transversal and co-graphic matroids in Section 3.2 and Section 3.3 respectively.

**Theorem 7.** *There is an absolute constant $\alpha \in (0,1)$ such that it is* PSPACE*-hard to approximate Transversal Bayesian Online Selection to a factor of $\alpha$.*

**Theorem 8.** *There is an absolute constant $\alpha \in (0,1)$ such that it is* NP*-hard to approximate Co-Graphic Bayesian Online Selection (*CGBOS*) to a factor of $\alpha$.*

Note that Theorem 8 only states NP-hardness. We also have as a corollary of Theorem 5 that CGBOS is PSPACE-complete.

## 3.1 Graphic Matroids

### 3.1.1 Hardness for Graphic Bayesian Online Selection

This subsection is devoted to proving Theorem 5. The proof techniques are inspired by the proof in [PPSW21] that shows the related online stochastic max-weight bipartite matching problem is PSPACE-hard. We first provide a stochastic variant of Max 2SAT and prove it is PSPACE-hard to approximate. Following this, we create an instance of GBOS that models the decisions that the optimal online algorithm for Stochastic Max 2SAT must make and show that the performance of the optimal algorithms for Stochastic Max 2SAT and GBOS on these instances are closely related.

#### PSPACE-Hard Problem: MAX-S2SAT

To prove Theorem 5, we first need some results about Stochastic Max 2SAT.

**Definition 3.** *In the* Stochastic Max 2SAT problem*, henceforth referred to as* MAX-S2SAT*, the input is a 2CNF formula $\phi$ over an ordered list of variables $(x_1, x_2, \ldots, x_n)$, where $n$*

*is even and for every $i = 1, \ldots, n-1$, either $x_i$ or $x_{i+1}$ is contained in some clause of $\phi$. We choose a value of* `True` *or* `False` *for $x_1$. Then, nature sets $x_2$ to either* `True` *or* `False` *with a probability of 0.5. We then get to choose the value of $x_3$, nature sets the value of $x_4$ to either* `True` *or* `False`*, and so on. Our goal is to maximize the expected number of satisfied clauses in $\phi$ after all the variables have been assigned a value. A variable is called a* random *variable if it is set by nature, and is called* deterministic *otherwise.*

The following lemma is crucial for our reduction, since it provides a problem to reduce from.

**Lemma 11.** *There exist absolute constants $k \in \mathbb{N}$ and $\alpha \in (0,1)$ so that it is* `PSPACE`*-hard to compute an $\alpha$-approximation for a* `MAX-S2SAT` *instance $\phi$ satisfying the requirement that each variable appears in at most $k$ clauses of $\phi$.*

The proof of Lemma 11 follows relatively directly from the analogous result for MAX-3SAT.

**Lemma 12** (Lemma 2.3 in [PPSW21])**.** *There exist absolute constants $k \in \mathbb{N}$ and $\alpha \in (0,1)$ such that it is* `PSPACE`*-hard to compute an $\alpha$-approximation for a* `MAX-S3SAT` *instance $\phi$ satisfying the requirement that each variable appears in at most $k$ clauses of $\phi$.*

To prove Lemma 12, we first need the following fact.

**Fact 1** (Fact 2.1 from [PPSW21])**.** *Let $Q, Q' \geq 0$ be positive quantities, such that $\frac{Q'}{Q} \leq \beta$, and let $\alpha \in (0,1)$. Then, an $(\frac{\alpha+\beta}{1+\beta})$-approximation to $Q + Q'$ yields an $\alpha$-approximation to $Q$.*

*Proof.* First note that

$$\frac{\alpha+\beta}{1+\beta} \geq \frac{\alpha + Q'/Q}{1 + Q'/Q} = \frac{\alpha \cdot Q + Q'}{Q + Q'}$$

because $\frac{\alpha+x}{1+x}$ is increasing for $\alpha \in (0,1)$ and positive $x$. Then our approximation for $Q+Q'$ yields a number in

$$\left[\frac{\alpha+\beta}{1+\beta} \cdot (Q+Q'), Q+Q'\right] \subseteq [\alpha \cdot Q + Q', Q+Q'] \,.$$

Subtracting $Q'$ results in a number in $[\alpha \cdot Q, Q]$ as wanted. □

While the proof of Lemma 11 is similar to the classic reduction for MAX-2SAT as found in [GJS74], we reproduce the details for completeness.

*Proof of Lemma 11.* Let $\phi'$ be an instance of `MAX-S3SAT` with variables $x_1, x_2, \ldots, x_n$ such that each variable appears in at most $k'$ clauses of $\phi'$. Let us construct an instance of `MAX-S2SAT` $\phi$ as follows:

- define $x_1, x_2, \ldots, x_n$ as variables of $\phi$
- if a clause in $\phi'$ contains at most two literals, then directly add this clause to $\phi$
- if $i$th clause in $\phi'$ contains three literals, i.e. is of the form $(\ell_1 \vee \ell_2 \vee \ell_e)$, define a new deterministic variable $c_i$, append it to the end of our current variables; after that add ten following *new clauses* to $\phi$

  $$(\ell_1), (\ell_2), (\ell_3), (c_i), (\neg\ell_1 \vee \neg\ell_2), (\neg\ell_2 \vee \neg\ell_3), (\neg\ell_1 \vee \neg\ell_3), (\ell_1 \vee \neg c_i), (\ell_2 \vee \neg c_i), (\ell_3 \vee \neg c_i)$$

First, note that $\phi$ satisfies the requirements of Lemma 11 for $k = 10k'$. Indeed, each clause has at most two variables, and each variable appears in at most $k$ clauses of $\phi$.

Consider any truth values assigned to literals $\ell_1, \ell_2, \ell_3$. If the values of $\ell_1$, $\ell_2$, $\ell_3$ satisfy $(\ell_1 \vee \ell_1 \vee \ell_3)$, then we can assign a value to $c_i$ so that 7 of our new clauses are satisfied. Note that there is no assignment of `True` and `False` to $c_i$ satisfying more than 7 new clauses. If the values of $\ell_1$, $\ell_2$, $\ell_3$ do not satisfy $(\ell_1 \vee \ell_2 \vee \ell_3)$, then we can satisfy at most 6 of our new clauses by any assignment to $c_i$; and we can satisfy exactly 6 by setting $c_i$ to `False`.

Let $OPT_{on}(\phi')$ denote the optimal expected value for $\phi'$, and equivalently for $OPT_{on}(\phi)$. We claim that $OPT_{on}(\phi) = 6m + OPT_{on}(\phi')$ where $m$ is the number of clauses with three literals in $\phi'$.

We can construct an algorithm for $\phi$ that mimics the decisions of $OPT_{on}(\phi')$ until it reaches clause variables, at which point it assigns them to greedily maximize remaining clauses. As we saw above, we gain 6 satisfied clauses in $\phi$ for each non-satisfied clause in $\phi'$ and 7 satisfied clauses in $\phi$ for each satisfied clause in $\phi$, giving

$$OPT_{on}(\phi) \geq 6m + OPT_{on}(\phi')\,.$$

Now construct an algorithm for $\phi'$ that mimics the decisions of $OPT_{on}(\phi)$. An optimal algorithm for $\phi$ always yields value at least 6 satisfied clauses per a three-literal clause in $\phi'$, since it can set the clause variables $c_i$, $i = 1, \ldots, m$ to get at least 6 satisfied clauses for each original three-literal clause of $\phi'$.

Further, as we saw earlier, it is impossible for $OPT_{on}(\phi)$ to gain more than 7 satisfied clauses for any given three-literal clause of $\phi'$. In particular, it can gain 7 exactly when

the values of $x_1$, $x_2$, ..., $x_n$ satisfy the clause from $\phi'$. So our constructed algorithm for $\phi$ satisfies $OPT_{on}(\phi) - 6m$ clauses, which combined with above implies the equality

$$OPT_{on}(\phi) = 6m + OPT_{on}(\phi')\,.$$

Now suppose that $\alpha' \in (0,1)$ is the constant from Lemma 12. In order to use Lemma 1, let $Q := OPT_{on}(\phi')$, $Q' := 6m$, and $\beta := 12$. Note that $OPT_{on}(\phi') \geq \frac{m}{2}$ and

$$\frac{Q'}{Q} \leq \frac{6m}{m/2} = \beta\,.$$

Now Lemma 1 implies an $\alpha := \frac{\alpha+12}{13}$ approximation for `MAX-S2SAT` yields an $\alpha'$-approximation for `MAX-S3SAT`. So it is `PSPACE`-hard to approximate `MAX-S2SAT` within a factor of $\alpha$. □

### Reduction from `MAX-S2SAT` to `GBOS`

Now, we can state our reduction from `MAX-S2SAT` to `GBOS`, represented pictorially in Figure 3.1. Let $\phi$ be an instance of `MAX-S2SAT` as in Lemma 11 with variables $(x_1, x_2, \ldots, x_n)$ and constant $k$. Let $m$ indicate the number of clauses in $\phi$. We construct an instance $\mathcal{I}_\phi$ of `GBOS` as follows.

The vertex set in the graph for $\mathcal{I}_\phi$ consists of a single central vertex $w$ and two vertices for each variable $x_i$, $i = 1, \ldots, n$ in $\phi$. We label the vertices for the variable $x_i$, $i = 1, \ldots, n$ as $x_i$ and $\neg x_i$

$$V := \{w\} \cup \{\, x_i,\, \neg x_i \,:\, 1 \leq i \leq n\}\,.$$

The edges of $\mathcal{I}_\phi$ arrive in three distinct phases and have distinct value distributions. Before specifying $\mathcal{I}_\phi$ in full detail, let us explain the intuition behind the construction. The first phase simulates the selection of truth values for variables in the same order as they appear in $\phi$. The second phase accounts for the number of clauses satisfied by the value selection of phase one. For this, the edges in the second phase correspond to clauses and attain very high values with very small probability, so that multiple clause edges attaining value is very unlikely. The third phase guarantees that for each variable at most one truth value is selected in the first phase by the optimal online policy.

**First Phase.** Respecting the order of variables as in $\phi$, two edges $wx_i$ and $w\neg x_i$ arrive for each variable $x_i$, $i = 1, \ldots, n$. The edge $wx_i$ arrives immediately before $w\neg x_i$. If $x_i$ is a deterministic variable, both edges $wx_i$ and $w\neg x_i$ have deterministic value 1. If $x_i$ is a random variable $x_i$, the edge $wx_i$ has value 2 with probability 0.5 and value 0 otherwise; the edge $w\neg x_i$ has deterministic value 1.

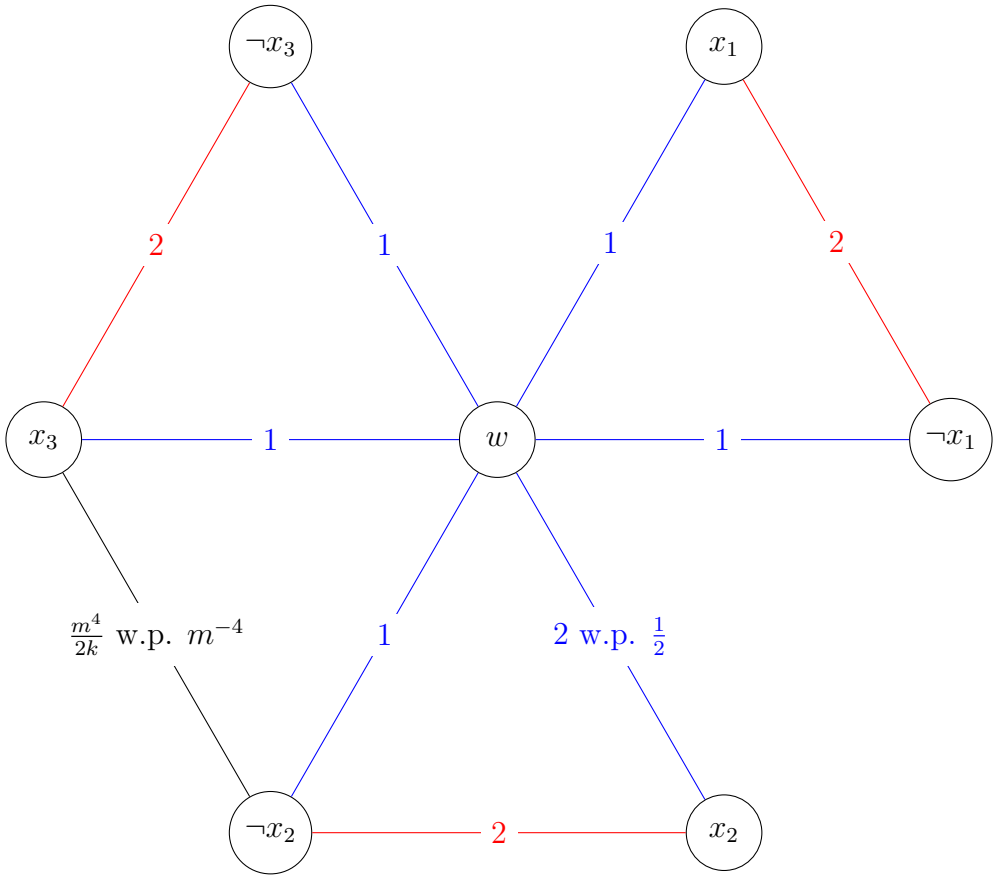



Figure 3.1: The produced GBOS instance from a MAX-S2SAT instance with 3 variables and a single $(x_2 \vee \neg x_3)$ clause. $x_1$ and $x_3$ are deterministic variables, while $x_2$ is a random variable. Phases 1, 2, and 3 consist of the blue, black, and red edges respectively.

**Second Phase.** An edge $\neg\ell_1\neg\ell_2$ arrives for each clause $(\ell_1 \vee \ell_2)$ in $\phi$. Each such edge has value $m^4/2k$ with probability $m^{-4}$, and value 0 otherwise. For each clause $(\ell)$ in $\phi$, i.e., for each clause with a single literal, an edge $w\neg\ell$ arrives with the same value distribution.

**Third Phase.** An edge $x_i\neg x_i$ arrives for each variable $x_i$, $i = 1, \ldots, n$. The edge $x_i\neg x_i$ has deterministic value 2.

Note that within the first phase, the order of arrivals is important. In contrast, the order of edge arrivals within the second or third phase can be arbitrary.

**Optimal Online Policy for $\mathcal{I}_\phi$**

Let $OPT_{on}(\phi)$ refer to the expected value of the optimal online algorithm for `MAX-S2SAT` on $\phi$. Let $ALG_{opt}$ denote an optimal algorithm for `GBOS`. We can assume that $ALG_{opt}$ selects only arrived edges. We say that an edge *arrived* if the value of the edge is nonzero.

**Lemma 13.** *Algorithm $ALG_{opt}$ selects exactly one of $wx_i$, $w\neg x_i$ for each variable $x_i$.*

*Proof.* First, we show that $ALG_{opt}$ selects at least one of $wx_i$, $w\neg x_i$ for each variable $x_i$. Suppose for the sake of contradiction that, after some history occurring with probability $q > 0$, $ALG_{opt}$ selects neither $wx_i$ nor $w\neg x_i$. Let $\mathcal{A}$ be an algorithm that matches the decisions of $ALG_{opt}$, except that it selects $w\neg x_i$, and afterwards does not select any edges that would then introduce a cycle.

Consider what we lose out on by running $\mathcal{A}$ instead of $ALG_{opt}$. Let $\mathcal{A}(\mathcal{I}_\phi)$ be the expected value obtained by $\mathcal{A}$ on $\mathcal{I}_\phi$. If no edge adjacent to $x_i$ or $\neg x_i$ arrive in the second phase, then the vertex $x_i$ has no adjacent selected edges before the third phase. So both $ALG_{opt}$ and $\mathcal{A}$ select the edge $x_i\neg x_i$ with value 2 in the third phase, since in this case $x_i\neg x_i$ does not lead to a cycle for any of them.

If some edge adjacent to $x_i$ or $\neg x_i$ arrives in the second phase, then $\mathcal{A}$ can miss out on at most one edge that $ALG_{opt}$ selects. The value of this missed edge can be at most $m^4/2k$. Here, we assume $m^4/2k \geq 2$, which is true for sufficiently large $m$. By the union bound, the probability that any clause edge adjacent to $x_i$ or $\neg x_i$ arrives is at most $k \cdot m^{-4}$, so we have

$$\mathcal{A}(\mathcal{I}_\phi) - OPT_{on}(\mathcal{I}_\phi) \geq q\left(1 - k \cdot m^{-4} \cdot \frac{m^4}{2k}\right) = \frac{q}{2} > 0\,,$$

contradicting the optimality of $ALG_{opt}$.

Second, note that $ALG_{opt}$ never selects both $wx_i$ and $w\neg x_i$ for any variable $x_i$, $i = 1, \ldots, n$. To see this fact, observe that it is more beneficial for $ALG_{opt}$ to accept only $wx_i$ in the first phase and to commit to accept $x_i\neg x_i$ in the third phase. □

The following lemma shows that $ALG_{opt}$ respects the random choice that nature made for the truth value of each random variable.

**Lemma 14.** *For a random variable $x_i$, algorithm $ALG_{opt}$ selects $wx_i$ if and only if $wx_i$ arrives.*

*Proof.* It is sufficient to show that if $wx_i$ arrives, i.e. has value 2, $ALG_{opt}$ selects $wx_i$. For the sake of contradiction, suppose that after some history occurring with probability $q > 0$, the edge $wx_i$ arrives but $ALG_{opt}$ does not select it.

Let $\mathcal{A}$ be an algorithm that matches the decisions of $ALG_{opt}$, except that it selects $wx_i$ when it arrives, and afterwards $\mathcal{A}$ does not select any edges that would then introduce a cycle. Analogously to the proof of Lemma 13 we obtain

$$\mathcal{A}(\mathcal{I}_\phi) - ALG_{opt}(\mathcal{I}_\phi) \geq q \left(1 - k \cdot m^{-4} \cdot \frac{m^4}{2k}\right) = \frac{q}{2} > 0\,.$$

contradicting the optimality of $ALG_{opt}$. □

Note that all arrived edges after phase 1 have non-increasing value, so by the greedy selection property of matroids, it is optimal to act greedily on them.

**Remark 1.** *After the first phase, $ALG_{opt}$ selects every edge that arrives and can be selected without introducing a cycle.*

**Remark 2.** *Note that the proofs of the above statements, i.e. Lemma 13, Lemma 14 and Remark 1, can be adapted to any online policy $ALG_{aprx}$ that attempts to maximize the expected total value of the selected elements but is limited to polynomial time computations. Indeed, the proofs of Lemma 13, Lemma 14 and Remark 1 show that if $ALG_{aprx}$ does not follow the rules outlined in these lemmas, one can compute an alternative online policy $\mathcal{A}$ that satisfies all the outlined rules, runs in polynomial time, and achieves a strictly better expected total value of the selected items.*

### Gains for $\mathcal{I}_\phi$

Let us consider the gains that are achieved by $ALG_{opt}$ and can be achieved by an online policy $ALG_{aprx}$ that runs in polynomial time and tries to maximize the expected total value of the selected elements. Due to the results in the previous section, we can assume that both $ALG_{opt}$ and $ALG_{aprx}$ follow the rules below:

1. exactly one of $wx_i$, $w\neg x_i$ is selected for each variable $x_i$
2. $wx_i$ is selected for a random variable $x_i$ if and only if $wx_i$ arrives
3. after the first phase, every edge is selected whenever the edge arrives and can be selected without introducing a cycle.

**Lemma 15.** *Let $\mathcal{A}$ be an online policy that follows rules (1), (2) and (3). Let $P_{\mathcal{A}}(\mathcal{I}_\phi)$ be the probability that a randomly selected clause in $\phi$ is satisfied if the* `True` *and* `False` *values for $x_i$, $i = 1, \dots, n$ were assigned according to the edges selected by $\mathcal{A}$ in the first phase. In particular, we assign $x_i$ the value* `True` *if $\mathcal{A}$ selects $wx_i$, and we assign* `False` *otherwise. Then, we have*

$$\mathcal{A}(\mathcal{I}_\phi) = 1.25n + 2n \cdot \left(1 - m^{-4}\right)^m + \left(2n + P_{\mathcal{A}}(\mathcal{I}_\phi) \cdot \left(\frac{m^4}{2k} - 2\right)\right) \cdot m \cdot m^{-4} \cdot \left(1 - m^{-4}\right)^{m-1} + \delta_{\mathcal{A}}\,,$$

*where $\delta_{\mathcal{A}}$ is a number in $[0, 2m^{-1}]$.*

Clearly, $P_{\mathcal{A}}(\mathcal{I}_\phi)$ in Lemma 15 is the number of clauses satisfied by such an assignment divided by the total number of clauses, i.e. by $m$.

*Proof of Lemma 15.* Let $\mathcal{A}_1(\mathcal{I}_\phi)$, $\mathcal{A}_2(\mathcal{I}_\phi)$ and $\mathcal{A}_3(\mathcal{I}_\phi)$ be the random variables indicating the total value of elements selected by $\mathcal{A}$ in each of the three phases. We have that $\mathbb{E}[\mathcal{A}_1(\mathcal{I}_\phi)] = 1.25n$ because in expectation each deterministic variable and randomized variable in $\phi$ lead to the gain of 1 and 1.5 in expectation by rules (1) and (2).

Let $C$ be the random variable indicating the number of edges that arrived in the second phase. We estimate the gain from $\mathcal{A}_2(\mathcal{I}_\phi)$ and $\mathcal{A}_3(\mathcal{I}_\phi)$ conditioned on three possible events $C = 0$, $C = 1$, and $C > 1$. Clearly, we have $\mathbb{E}[\mathcal{A}_2(\mathcal{I}_\phi) + \mathcal{A}_3(\mathcal{I}_\phi) \mid C = 0] = 2n$, as under the condition $C = 0$ there is no gain in the second phase and every edge is selected in the third phase by rule (3). We also have

$$\mathbb{E}[\mathcal{A}_2(\mathcal{I}_\phi) + \mathcal{A}_3(\mathcal{I}_\phi) \mid C = 1] = 2n + P_{\mathcal{A}}(\mathcal{I}_\phi) \cdot \left(\frac{m^4}{2k} - 2\right)\,,$$

because in this case the edge that arrived in the second phase is selected exactly when the corresponding clause is satisfied by the `True` and `False` assignment produced by the first phase, while the clause is satisfied with probability $P_{\mathcal{A}}(\mathcal{I}_\phi)$.

Let us state a trivial bound for the gain in the second and third phases conditioned on $C > 1$

$$\mathbb{E}[\mathcal{A}_2(\mathcal{I}_\phi) + \mathcal{A}_3(\mathcal{I}_\phi) \mid C > 1] \leq 2n + m \cdot \frac{m^4}{2k} \leq 2 \cdot \frac{m^5}{2k} \leq m^5\,,$$

which is obtained by assuming that all edges in the last two phases arrive and $\mathcal{A}$ selects all of them. However, the event that $C > 1$ has a very small probability of occurring

$$\mathbf{Pr}[C > 1] = \sum_{t=2}^{m} \binom{m}{t} \left(m^{-4}\right)^t \left(1 - m^{-4}\right)^{m-t} \leq \sum_{t=2}^{m} m^t \left(m^{-4t}\right) \leq 2m^{-6}\,.$$

So we have

$$\delta_{\mathcal{A}} := \mathbb{E}[\mathcal{A}_2(\mathcal{I}_\phi) + \mathcal{A}_3(\mathcal{I}_\phi) \mid C > 1] \cdot \mathbf{Pr}[C > 1] \in [0, 2m^{-1}] .$$

Thus, the expected gain of $\mathcal{A}$ can be estimated as follows

$$\mathcal{A}(\mathcal{I}_\phi) = \mathbb{E}[\mathcal{A}_1(\mathcal{I}_\phi)] + \mathbb{E}[\mathcal{A}_2(\mathcal{I}_\phi) + \mathcal{A}_3(\mathcal{I}_\phi)] = 1.25n + 2n \cdot \left(1 - m^{-4}\right)^m + \left(2n + P_{\mathcal{A}}(\mathcal{I}_\phi) \cdot \left(\frac{m^4}{2k} - 2\right)\right) \cdot m \cdot m^{-4} \cdot \left(1 - m^{-4}\right)^{m-1} + \delta_{\mathcal{A}} ,$$

where we used that $\mathbf{Pr}[C = 0] = (1 - m^{-4})^m$ and $\mathbf{Pr}[C = 1] = m \cdot m^{-4} \cdot (1 - m^{-4})^{m-1}$. □

**Putting Everything Together**

Let us define $\gamma := (1 - m^{-4})^{m-1}$ and notice that the limit of $\gamma$ as $m$ goes to infinity equals 1. Recall that $k$ is an absolute constant. Thus, considering the expected gain of an online policy, we can conclude that when $\mathcal{A}$ is $ALG_{opt}$, the value of $P_{\mathcal{A}}$ is $OPT_{on}(\phi)/m$. Let us now rewrite the expected gain of the optimal policy $ALG_{opt}$ as follows

$$\underbrace{1.25n + 2n \cdot \gamma + 2n \cdot m^{-3} \cdot \gamma + \delta_{ALG_{opt}}}_{=:Q'} + \underbrace{OPT_{on}(\phi) \cdot \left(\frac{m^4}{2k} - 2\right) \cdot m^{-4} \cdot \gamma}_{=:Q} .$$

Let us define $Q$ and $Q'$ as above and define $\beta := 54k/\gamma$.

To use Fact 1, we need to show that $Q'/Q \leq \beta$. Notice that $n \leq 4m$ because no consecutive pair of variables is missing from all clauses. Also, $OPT_{on}(\phi) \geq \frac{m}{2}$ because we can achieve $\frac{m}{2}$ satisfied clauses with a random variable assignment. So we obtain $n \leq 8 \cdot OPT_{on}(\phi)$. Thus, we have

$$Q' \leq 3.25n + 0.25 + \delta_{ALG_{opt}} \leq 26 \cdot OPT_{on}(\phi) + 0.5 \leq 27 \cdot OPT_{on}(\phi)$$

for sufficiently large $m$. Now we have that

$$Q'/Q \leq (27 \cdot OPT_{on}(\phi)) \,/\, (0.25 \cdot \gamma \cdot OPT_{on}(\phi)/k) = \beta$$

for sufficiently large $m$. Also, since $k$ is an absolute constant, we have that $\beta = 108k/\gamma$ can be seen as a constant. So, we can apply Fact 1 and find that an $\alpha' := \frac{\alpha + 108k/\gamma}{1 + 108k/\gamma}$ approximation for `GBOS` gives an $\alpha$ approximation for `MAX-S2SAT`. Thus, by Lemma 11 there is some constant $\alpha' \in (0, 1)$ such that it is `PSPACE`-hard to approximate `GBOS` to a factor of $\alpha'$.

### 3.1.2 No PTAS-Compatible Distribution-Agnostic Order

Let us show that for a graphic matroid on a complete graph, there is no PTAS-compatible distribution-agnostic order. Let us be given a complete graph $G = (V, E)$ with $|V| = 3n + 2m + 1$, where $n$ is even, and let us be given some arrival order for $E$. Let us show how one can construct a reduction for `MAX-S2SAT` analogously to Section 3.1.1.

**Edges for Third Phase.** In the reverse order of arrival, greedily construct a matching $A := \{a_1, \ldots, a_n\}$ such that $|A| = n$. Each edge in $A$ has deterministic value 2.

**Edges for Second Phase.** In the reverse order of arrival, after constructing $A$, greedily keep adding edges to $B := \{b_1, \ldots, b_m\}$ such that $A \cup B$ is a matching and $|B| = m$. Each edge in $B$ has value $m^4/2k$ with probability $m^{-4}$, and value 0 otherwise.

**Edges for First Phase.** Consider the vertex set $U$, which are the vertices not matched by the matching $A \cup B$. Consider a vertex $w \in U$ and let $v_1$, $\overline{v}_1$, $v_2$, $\overline{v}_2$, $\ldots$, $v_n$, $\overline{v}_n$ be such that the edges $wv_1$, $w\overline{v}_1$, $wv_2$, $w\overline{v}_2$, $\ldots$, $wv_n$, $w\overline{v}_n$ are the first $n$ edges adjacent to $w$ and their arrivals are in the above order. If $i$ is odd, both edges $wv_i$ and $w\overline{v}_i$ have deterministic value 1. If $i$ is even, the edge $wv_i$ has value 2 with probability 0.5 and value 0 otherwise; and the edge $w\overline{v}_i$ has deterministic value 1.

**Auxiliary Edges.** For each $i = 1, \ldots, n$, both edges $v_i\alpha_i$ and $\overline{v}_i\gamma_i$ have deterministic value 3, where $\alpha_i$ and $\gamma_i$ are the endpoints of $a_i$. For each $j = 1, \ldots, m$ both edges $u_j\beta_j$ and $l_j\tau_j$ have deterministic value 3, where $\beta_j$, $\tau_j$ are the endpoints of $b_j$ and $l_j$, $u_j$ correspond to "negations" of the literals in the $j$th clause of $\phi$.

Every remaining edge has deterministic value 0. Note, that all auxiliary edges arrive before the edges in the second and third phase due to the greedy construction of $A$ and $B$. Also, all auxiliary edges are selected by an optimal online policy.

The same analysis as in Section 3.1.1, but with a slightly degraded inapproximability constant due to the additional guaranteed gain from the auxiliary edges.

## 3.2 Transversal Matroids

**Definition 4.** *Given a bipartite graph $G = (L \cup R, E)$, the* **transversal matroid** *of $G$ is the matroid $\mathcal{M} = (L, \mathcal{I})$ where a set $A \subseteq L$ is independent if and only if there is a matching in $G$ that saturates $A$.*

The Transversal Bayesian Online Selection (`TBOS`) problem is the matroid Bayesian online selection problem restricted to transverasl matroids. In this section, we

show that one can embed essentially the exact same construction as we used for graphic matroids into transversal matroids, proving Theorem 7.

### 3.2.1 Reduction from MAX-S2SAT to TBOS

Now, we can state our reduction from MAX-S2SAT to TBOS. Let $\phi$ be an instance of MAX-S2SAT as in Lemma 11 with variables $(x_1, x_2, \ldots, x_n)$ and constant $k$. Let $m$ indicate the number of clauses in $\phi$. We construct an instance $\mathcal{I}_\phi$ of TBOS as follows.

Define the vertices of the bipartite graph $G = (L \cup R, E)$ as

$$L := \{x_i, \neg x_i, z_i \,:\, 1 \leq i \leq n\} \cup \{c_i \,:\, 1 \leq i \leq m\}$$

and

$$R := \{y_i, \neg y_i \,:\, 1 \leq i \leq n\}.$$

We first introduce edges to control the selection of variable vertices

$$E_1 := \{x_i y_i, \neg x_i \neg y_i \,:\, 1 \leq i \leq n\} \cup \{z_i y_i, z_i \neg y_i \,:\, 1 \leq i \leq n\}.$$

Next, we introduce edges to control the selection of clause vertices

$$E_2 := \{c_i \neg \ell_1, c_i \neg \ell_2 \,:\, \text{for the } i\text{th clause } (\ell_1 \vee \ell_2) \text{ in } \phi\}$$

and set $E := E_1 \cup E_2$.

As in our hardness result for graphic matroids, the vertices in $L$ arrive in three distinct phases:

**First Phase.** Respecting the order of variables as in $\phi$, two vertices $x_i$ and $\neg x_i$ arrive for each variable $x_i$, $i = 1, \ldots, n$, with $x_i$ arriving immediately before $\neg x_i$. If $x_i$ is a deterministic variable, both vertices $x_i$ and $\neg x_i$ have deterministic value 1. If $x_i$ is a random variable, then $x_i$ has value 2 with probability 0.5 and value 0 otherwise; $\neg x_i$ has deterministic value 1.

**Second Phase.** The clause vertex $c_i$ arrives for each clause in $\phi$. Each such vertex has value $m^4/2k$ with probability $m^{-4}$, and value 0 otherwise.

**Third Phase.** The vertex $z_i$ arrives for each variable $x_i$, $i = 1, \ldots, n$ with deterministic value 2.

Let $G' = (V', E')$ indicate the graph from the GBOS instance produced from $\phi$ as in Section 3.1. Define the bijection $g : L \to E'$ as follows. For each literal $\ell \in \{x_1, \neg x_1, \ldots, x_n, \neg x_n\}$,

let $g(\ell) := w\ell$. For each $z_i \in L$, let $g(z_i) := x_i \neg x_i$. Finally, for each clause vertex $c_i \in L$ arising from the $i$th clause $(\ell_1 \vee \ell_2) \in \phi$, let $g(c_i) := \neg\ell_1 \neg\ell_2$. Now for any set $S \subseteq L$ with at most one clause vertex $|S \cap \{c_1, \ldots, c_m\}| \leq 1$, it is clear that $S$ is independent if and only if $\{g(v) : v \in S\} \subseteq E'$ is independent.

Following this observation, the exact same analysis as in Section 3.1 can be applied, showing that

1. exactly one of $x_i$, $\neg x_i$ is selected for each variable $x_i$,
2. $x_i$ is selected for a random variable $x_i$ if and only if $x_i$ has non-zero value, and
3. after the first phase, every vertex is selected whenever the vertex has non-zero value and can be selected without breaking independence.

The exact same analysis of the optimal algorithm on $\mathcal{I}_\phi$ compared to the optimal algorithm on $\phi$ follows, yielding Theorem 7.

## 3.3 Co-Graphic Matroids

As an immediate consequence of Graphic Bayesian Online Selection being PSPACE-complete (Theorem 5), we obtain that Co-Graphic Bayesian Online Selection (CGBOS) is PSPACE-complete. Intuitively, this can be seen by viewing CGBOS as the problem of choosing a minimum-weight spanning tree to not select, while GBOS is the problem of choosing a maximum-weight spanning tree to select.

More rigorously, let $G = (V, E)$ denote an instance for GBOS, and let $v_{\max}$ denote the maximum possible value of any item in the instance. Then we can produce a CGBOS instance by updating all possible values $v$ to $v' := v_{\max} - v$. Letting $OPT$ denote the optimal expected revenue for the GBOS instance and $OPT'$ denote the optimal expected revenue for the produced CGBOS instance, it can be shown that

$$OPT' = OPT + v_{\max} \cdot (|E| - |V| + 1) - \sum_{e \in E} \mathbb{E}[v_e].$$

To show difficulty of approximating CGBOS as in Theorem 8, we reduce from a special case of the Tree Augmentation Problem.

**Definition 5.** *In the Tree Augmentation Problem (TAP), the input is a tree $T = (V, E_0)$, an additional set of edges $E' \subseteq V \times V$ such that $(V, E \cup E')$ is two edge connected and a cost function $c : E' \to \mathbb{N}$. The output is the minimum cost $\sum_{e \in E_1} c(e)$ of a set of new edges $E_1 \subseteq E'$ such that $(V, E_0 \cup E_1)$ is two edge connected.*

This problem has been well-studied in the approximation algorithms literature for many decades. Importantly for this section, it is known to be NP-hard to approximate.

**Lemma 16** (Theorem 4.1 from [KKL04], Theorem 2 from [FJ81])**.** *There is an absolute constant $\alpha > 1$ such that it is NP-hard to approximate TAP to a factor of $\alpha$. Further, there exists a constant $k \in \mathbb{N}$ such that this holds for instances of TAP in which the range of $c$ is $\{1, 2\}$, $|E'| \leq k \cdot |V|$, and the optimal answer is guaranteed to be at least $\frac{|V|}{k}$.*

Let $(T, c)$ be an instance of TAP as described in Lemma 16. We now state our reduction.

Let $G = (V, E \cup E')$ and $n := |V|$. We first make the edges of $E'$ arrive in arbitrary order. Each edge $e \in E'$ arrives with deterministic value $c(e)$. Following this, we make the edges of $E$ arrive in arbitrary order. Each edge of $E$ takes on value $3 \cdot k \cdot n^5$ with probability $n^{-5}$ and value 0 otherwise.

The intuition behind this instance is that a single edge in $E$ will be selectable exactly when it is not a bridge in $(V, (E' \setminus C) \cup E)$, where $C$ is the set of edges selected from $E'$. For this to be true for all edges in $E$, we wish to choose a maximum value $C$ such that $(V, (E' \setminus C) \cup E)$ has no bridges - equivalently, we wish to choose a minimum value $E' \setminus C$ satisfying this. Let $ALG_{opt}$ denote the optimal algorithm for our produced CGBOS instance and $OPT$ denote its expected gain. In the remainder of this section, we will study the behavior of $ALG_{opt}$ in order to characterize its expected gain.

**Remark 3.** *$ALG_{opt}$ will greedily select feasible edges in $E$ with non-zero value.*

Remark 3 is true because all edges in $E$ have the same value when non-zero, and so this rule is optimal by the greedy selection property of matroids. Noting this, let $f(A)$ be a random variable denoting the gain of $ALG_{opt}$ on $E$ after selecting $A \subseteq E'$. Using Remark 3 and the fact that it's incredibly unlikely for two edges in $E$ to both have a non-zero value, we can almost fully compute the expected value of $f$.

**Lemma 17.** *Let $A \subseteq E'$ and $\ell$ denote the number of non-bridge edges in $(V, (E' \setminus A) \cup E)$. Then we have that*

$$\mathbb{E}[f(A)] = 3 \cdot k \cdot \ell \cdot (1 - n^{-5})^{n-2} + \delta_A,$$

*where $\delta_A \in [0, 2 \cdot n^{-1}]$.*

*Proof.* Let $C$ be the random variable indicating the number of edges from $E$ that take on a non-zero value. We separate into cases based on the value of $C$.

When $C = 0$, we have that $\mathbb{E}[f(A) \mid C = 0] = 0$, as there are no edges to select. When $C = 1$, we have that we can select an edge exactly when said edge is not a bridge in $(V, (E' \setminus A) \cup E)$, so

$$\mathbb{E}[f(A) \mid C = 1] = \frac{\ell}{n-1} \cdot 3 \cdot k \cdot n^5$$

and

$$\mathbf{Pr}[C = 1] = (n-1) \cdot n^{-5} \cdot (1 - n^{-5})^{n-2}.$$

We bound the $C > 1$ case by showing that the probability of this event is incredibly low.

$$\begin{aligned}
\mathbf{Pr}[C > 1] &= \sum_{t=2}^{n-1} \binom{n-1}{t} \cdot n^{-5t} \cdot (1 - n^{-5})^{n-1-t} \\
&\leq \sum_{t=2}^{n-1} n^t \cdot n^{-5t} \\
&\leq 2 \cdot n^{-8}.
\end{aligned}$$

And we have that

$$\mathbb{E}[f(A) \mid C > 1] \leq (n-1) \cdot 3 \cdot k \cdot n^5 \leq n^7$$

for sufficiently large $n$.

These values can be used to bound

$$\mathbb{E}[f(A)] \geq 3 \cdot k \cdot \ell \cdot (1 - n^{-5})^{n-2}$$

and

$$\mathbb{E}[f(A)] \leq 3 \cdot k \cdot \ell \cdot (1 - n^{-5})^{n-2} + 2 \cdot n^{-1}.$$

□

Lemma 18 tells us that we can increase the number of selectable elements in $E$ without decreasing the number of elements we select in $E'$ by too much. This in turn implies the optimal solution will always allow every element in $E$ to be selected.

**Lemma 18.** *Let $A \subseteq E'$ and $b$ denote the number of bridges in $(V, (E' \setminus A) \cup E)$. Then there is a set $A' \subseteq A$ such that $|A'| \geq |A| - b$ and $(V, (E' \setminus A) \cup E)$ has no bridges.*

*Proof.* Let $e$ be any edge that is a bridge in $(V, (E' \setminus A) \cup E)$. Note that $e \in E$ because $E$ is a spanning tree. Consider the cut induced by removing $e$ in $(V, (E' \setminus A) \cup E)$. Because $(V, E \cup E')$ is two edge connected, there is some $f \in E'$ crossing the cut. So we have that $e$ is not a bridge in $(V, (E' \setminus (A \setminus \{f\})) \cup E)$.

Let $A' := A \setminus \{f\}$. If there are no bridges in $(V, (E' \setminus A') \cup E)$, we are done. Otherwise, repeat this process inductively with $A'$. In each iteration, there is at least one less bridge in the resulting graph, and only one edge is removed from $A'$. □

**Lemma 19.** *$ALG_{opt}$ will always select some $A \subseteq E'$ such that $(V, (E' \setminus A) \cup E)$ has no bridges.*

*Proof.* Suppose otherwise, that $(V, (E' \setminus A) \cup E)$ has $b > 0$ bridges. Let $A'$ be as in Lemma 18. Consider an algorithm that selects $A'$ and proceeds greedily. Label its expected gain as $ALG := f(A') + \sum_{e \in A'} c(e)$.

By Lemma 17, we have that

$$\begin{aligned} ALG - OPT &= f(A') - f(A) - \sum_{e \in A \setminus A'} c(e) \\ &\geq f(A') - f(A) - 2 \cdot b \\ &\geq 3 \cdot k \cdot (1 - n^{-5})^{n-2} \cdot ((n-1) - (n-1-b)) - 2 \cdot n^{-1} - 2 \cdot b \\ &= b \cdot (3 \cdot k \cdot (1 - n^{-5})^{n-2} - 2) - 2 \cdot n^{-1} \\ &> 0 \end{aligned}$$

for sufficiently large $n$. This contradicts the optimality of $ALG_{opt}$. □

**Corollary 4.** *Let $OPT_{\texttt{TAP}}$ denote the optimal answer for the* `TAP` *instance $(T, c)$. The expected gain of $ALG_{opt}$ is*

$$OPT = \sum_{e \in E'} c(e) - OPT_{\texttt{TAP}} + 3 \cdot k \cdot (n-1) \cdot (1 - n^{-4})^{n-2} + \delta,$$

*where $\delta \in [0, 2 \cdot n^{-1}]$.*

Now we can use Corollary 4 to show how a good estimate for `CGBOS` produces a good estimate for `TAP`.

**Lemma 20.** *For $\epsilon \in (0, 1)$, a polynomial time $(1-\epsilon)$-approximation for* `CGBOS` *can be used to attain a polynomial time $(1 + 5k^2 \cdot \epsilon)$-approximation for* `TAP`.

*Proof.* Let $\mathcal{B}$ be the result of running the approximation algorithm on our produced CGBOS instance. We have that $\mathcal{B} \in [(1-\epsilon)\cdot OPT, OPT]$. Let

$$X := \sum_{e \in E'} c(E) + 3 \cdot k \cdot (n-1) \cdot (1-n^{-4})^{n-2}.$$

Note that $X$ can be computed in polynomial time, as it only uses basic facts about the input graph. Return $\mathcal{A} := -\mathcal{B} + X + 1$ as our estimate for $OPT_{\texttt{TAP}}$. We claim this is a "good" estimate as described in the statement of this lemma.

First bound $\mathcal{A}$ from above as

$$\begin{aligned}
\mathcal{A} &= -\mathcal{B} + X + 1 \\
&\leq -(1-\epsilon)(X + \delta - OPT) + X + 1 \\
&\leq (1-\epsilon)\cdot OPT + \epsilon \cdot X + 1 \\
&\leq (1-\epsilon)\cdot OPT + \epsilon \cdot 5k^2 \cdot OPT + \frac{k}{n} \cdot OPT \\
&= (1 + \epsilon \cdot (5k^2 + \frac{k}{n} - 1)) \cdot OPT \\
&\leq (1 + 5k^2 \cdot \epsilon) \cdot OPT
\end{aligned}$$

for sufficiently large $n$. Here we used that $OPT \geq \frac{n}{k}$ and $|E'| \leq k \cdot n$.

Now bound $\mathcal{A}$ from below as

$$\begin{aligned}
\mathcal{A} &= -\mathcal{B} + X + 1 \\
&\geq OPT - \delta + 1 \\
&\geq OPT.
\end{aligned}$$

□

Let $\alpha' > 1$ be the inapproximability constant for TAP from Lemma 16. Then by Lemma 20, it is NP-hard to approximate CGBOS to a factor of $\alpha := 1 - \frac{\alpha'-1}{5k^2}$. This finishes the proof of Theorem 8.

# Chapter 4

# Prophet Secretary

In the prophet secretary setting, the arrival order of elements is selected uniformly randomly among all permutations of the elements, and is not revealed ahead of time to the gambler. In [DGR+23], a PTAS is given in the single-item case.

This setting is substantially more complicated than the prophet inequality setting discussed in the rest of this thesis. While the prophet inequality setting has a very simple optimal algorithm in the single-item case, it's not known if the single-item case is even solvable in polynomial time in the prophet secretary setting. The core difficulty that kills many techniques used in the prophet inequality setting is that, because the order of the arriving elements is not revealed to the gambler ahead of time, one must keep track of which elements have been seen, and thus which elements are left, in order to have a full state. In the prophet inequality setting, this can be represented with a single integer representing the timestamp one is at. But in the prophet secretary setting, there are $2^n$ possibilities for sets of observed elements — far too many to enumerate.

The PTAS for the single-item case in [DGR+23] *discretizes* the values and probabilities in the distribution supports. After some pre-processing and dealing with very low probabilities via a clever front-loading technique, there are only a constant number of "types" of elements left. Then the number of observed elements of each type can be kept track of in polynomial time.

In this chapter, we use new ideas to give a PTAS for all constant-rank matroids, subject to the constraint that all probabilities in all element's distributions are lower-bounded by some constant. We refer to this constraint on probabilities as "high-probability". While the discretization and preprocessing steps used in [DGR+23] quickly yield a PTAS in this high-probability case for the single-item setting, new ideas are needed to provide an algorithm

for all constant-rank matroids, and our final algorithm bears little resemblance to the one given in [DGR+23] beyond the initial discretization and preprocessing.

## 4.1 PTAS for High-Probability Constant-Rank Matroids

This section is devoted to the proof of the following theorem.

**Theorem 9.** *There is a PTAS for prophet inequality on matroids of constant rank $r$ in which every element $u_i$ attains each of its support values with probability at least some constant $p > 0$.*

The PTAS proceeds by first discretizing all support values of the item's distributions to powers of $(1 + \epsilon)$. After discarding small value elements, we obtain a nearly-identical instance with only constantly many support values.

From here, we process elements by decreasing order of their value $v$. Roughly speaking, if there is a sufficient number of elements with support value $v$ in any part of the matroid, then we will almost certainly be able to form a basis of this part with each element having value $v$ or higher. In the rest of the matroid, there can only be a constant number of elements with support value $v$ due to the lower-bound $p$ on realizing $v$ and the upper-bound on the rank. The proof proceeds by showing that we can just keep track of these special elements for each possible support value, and either ignore or take for granted all other elements. Since we only keep track of a constant number of elements, this can be accomplished with a straightforward dynamic programming solution in polynomial time.

### 4.1.1 Preprocessing

In this subsection, we show that we can preprocess any instance into one that has at most a constant number of distinct support values, with only a small loss in the optimal value.

**Remark 4.** *Letting $OPT_{\mathcal{M}}$ denote the optimal online gain on the prophet secretary instance $\mathcal{M}$, we can assume without loss of generality that $OPT_{\mathcal{M}} \in [1 - 1/e, 1]$ by dividing all values by $OPT$, the expected optimal offline (i.e., prophet) value. This is due to the lower-bound of $1 - 1/e$ on the competitive ratio for matroid prophet secretary given in [EHKS18].*

*We further note that $OPT$ can be computed in polynomial time with a straightforward dynamic programming algorithm.*

From now on, we assume that $OPT_{\mathcal{M}} \in [1 - 1/e, 1]$ for our instance $\mathcal{M}$.

**Lemma 21.** *For any $\epsilon_0 \in (0, \min\{p, \frac{1}{r}\})$ and instance $\mathcal{M}$ of matroid prophet secretary with rank $r$ and probability lower-bound $p$, one can compute in polynomial time an instance $\mathcal{M}'$ such that*

- *$\mathcal{M}'$ has $O(\log_{1+\epsilon_0}(\epsilon_0^{-1})) \leq O(\epsilon_0^{-2})$ support values,*
- *the rank of $\mathcal{M}'$ is $r$,*
- *$\mathcal{M}'$ has probability lower-bound $p$, and*
- *$OPT_{\mathcal{M}'} \in [1 - O(\epsilon_0), 1] \cdot OPT_{\mathcal{M}}$.*

*Proof.* We construct $\mathcal{M}'$ from $\mathcal{M}$ by rounding down support values on each element of $\mathcal{M}$. First, discard (i.e., set the probability of realization to 0) all small support values $v < \frac{\epsilon_0}{r}$. Because at most $r$ elements may be selected from the matroid, this discarding step yields an additive loss of at most $\epsilon_0$, which due to our bounds on $OPT$ is at most an $O(\epsilon_0)$ multiplicative factor.

Making the assumption that there are no loop elements, each support value $v$ is upper-bounded by $\frac{1}{p}$ or else its expected value, and thus $OPT_{\mathcal{M}}$, would exceed 1. So all support values lie in the range $[\frac{\epsilon_0}{r}, \frac{1}{p}]$. Due to our stated assumptions on $\epsilon_0$, we have that all support values of $\mathcal{M}'$ lie in $[\epsilon_0^2, \epsilon_0^{-1}]$.

Finally, we round down all remaining support values of $\mathcal{M}'$ to the nearest power of $(1 + \epsilon_0)^k$, losing a factor of at most $(1 - \frac{1}{1+\epsilon_0}) \cdot OPT \leq O(\epsilon_0) \cdot OPT$. Due to our bounded value range of $[\epsilon_0^2, \epsilon_0^{-1}]$, the only possible values of $k$ are in $\left[\lfloor -2 \log_{1+\epsilon_0}(\epsilon_0^{-1}) \rfloor, \log_{1+\epsilon_0}(\epsilon_0^{-1})\right]$. □

Due to Lemma 21, we will assume from now on that our input instances have only a constant number of elements. Note that this constant depends on the amount of error $\epsilon_0$ we are willing to introduce, but this constant $\epsilon_0$ can be adjusted as a first step independently of the following subsections. Since we consider only approximate solutions, this adds only an $O(\epsilon_0)$ multiplicative factor to our loss.

### 4.1.2 Warm-up: The Single-Item Case

We begin with the single-item, i.e., rank-1 matroid case.

Recall that there is a constant amount of nonzero values $w_1 \geq w_2 \geq \ldots \geq w_k$ that elements can take on. Begin by grouping the elements of $\mathcal{M}$ according to their support values. Recursively define $G(w_j)$ beginning with $j = 1$, followed by increasing $j$. Let $G(w_1)$ be all elements $u_i \in U$ such that $\mathbf{Pr}[v_i = w_1] \neq 0$. For $j \geq 2$, define $G(w_j)$ as the set of elements $u_i \in U$ such that

1. $u_i \notin G(w_t)$ for all $1 \leq t < j$ and
2. $\mathbf{Pr}[v_i = w_j] \neq 0$.

In other words, $G(w_j)$ is the set of elements in $U$ with largest support value $w_j$. Together, these groups partition $U$.

Notice that the probability that an element of $G(w_1)$ takes on value $w_1$ is at least $1-(1-p)^{|G(w_1)|}$, as all support values have probability at least $p$. So if $|G(w_1)| \geq \log_{1-p}(\epsilon)$, there is at least a $(1-\epsilon)$ probability that at least one value of $w_1$ is realized. If this is the case, then the optimal answer will be very close to $w_1$. Otherwise, if $|G(w_1)| < \log_{1-p}(\epsilon)$, then there is only a constant number of elements in $G(w_1)$, which can be kept track of with a dynamic programming type solution.

We follow this pattern of thought for the remaining values $w_j$. If $|G(w_j)|$ is sufficiently large, then it is incredibly likely that there is an element of $G(w_j)$ that takes on value $w_j$. Further, by adjusting our threshold, we can assume this element arrives *after* all elements of $\cup_{t<j} G(w_t)$. So if $|G(w_j)|$ is sufficiently large, we can instead play the game where we are guaranteed to be allowed to select an item with value $w_j$ at the conclusion of the game. While this game may have a higher expected reward than the original game, its expected reward will not differ by very much, as the original game almost always has an essentially equivalent option. Then we can ignore all elements outside of $\cup_{t<j} G(w_t)$, keeping track of these special items with a dynamic programming solution. After all these elements have been observed, we then award ourselves a value of $w_j$, as we know this will almost certainly be an option.

For any prophet secretary instance $U'$, let $A_{exact}(U', w)$ be an algorithm that takes in $U'$ and some weight $w$, and outputs the expected optimal prophet secretary gain assuming that $w$ may be taken freely after observing the realization of all elements in $U'$. Such an algorithm can be implemented with a simple exponential-size dynamic program - we will only use it for constant-size $U'$, for which it runs in constant time.

We claim that Algorithm 3 is a PTAS for prophet secretary instances in which there are a constant number of values $k$ and a constant lower-bound $p$ on all support probabilities.

**Algorithm 3:** RankOneProphetSecretary$(U, k, p, \epsilon)$

Label $w_j$ and $G(w_j)$ as in items 1 and 2 for $1 \leq j \leq k$ and let $w_{k+1} \leftarrow 0$ ;
$t \leftarrow k+1$ ;
$\gamma_0 \leftarrow 0, \alpha_0 \leftarrow 0$ ;
**for** $j = 1, 2, \ldots, k$ **do**
    $\gamma_j \leftarrow \gamma_{j-1} + \alpha_{j-1}$ ;
    $\alpha_j \leftarrow \lceil \log_{1-p/(\gamma_j+1)}(\epsilon) \rceil$ ;
    **if** $|G(w_j)| \geq \alpha_j$ **then**
        $t \leftarrow j$ ;
        break ;
    **end**
**end**
$U' \leftarrow \bigcup_{j<t} G(w_j)$ ;
Return $\mathcal{A}_{exact}(U', w_t)$ ;

Note that $\alpha_j$ depends only on $j$, $p$, and $\epsilon$. Therefore, when $k$, $p$, and $\epsilon$ are constant, all $\alpha_j$ are bounded by some constant. We have that $|U'| < \sum_{j=1}^{k} \alpha_j$, so $U'$ is of constant size and thus $\mathcal{A}_{exact}$ will run in constant time. This implies that Algorithm 3 runs in polynomial time, as desired.

Now we need only analyze the correctness of the algorithm.

**Theorem 10.** *For an instance $(U, k, p)$ of single-item prophet secretary, let $OPT$ denote the expected philosopher's gain, and $ALG := RankOneProphetSecretary(U, k, p, \epsilon)$ be the result of running Algorithm 3 on the instance. Then we have that $OPT \in [(1-\epsilon) \cdot ALG, ALG]$.*

To show that Theorem 10 holds, we first need the following bound on the probability that the outside option $w_t$ is actually available.

**Lemma 22.** *If $t \neq k+1$, then the probability that at least one element $u \in G(w_t)$ arrives after all elements of $U'$ and takes on value $w_t$ is at least $1-\epsilon$.*

*Proof.* Consider any $u \in G(w_t)$. The probability that $u$ arrives after all elements of $U'$ is $\frac{1}{|U'|+1} \geq \frac{1}{\gamma_t+1}$. Further, the probability that $u$ takes on value $w_t$ is at least $p$, independently

of where it arrives. So with probability at least $\frac{p}{\gamma_t+1}$, $u$ arrives after all elements of $U'$ and takes on value $w_t$.

In fact, this holds for all elements of $G(w_t)$ independently, so the probability that none arrive after $U'$ and take on value $w_t$ is at most

$$\left(1 - \frac{p}{\gamma_t + 1}\right)^{\alpha_j} = \epsilon.$$

□

*Proof of Theorem 10.* Observe that $ALG \geq OPT$, as the only options that $ALG$ does not have access to are those of weight less than $w_t$. Because $ALG$ always has the option of earning $w_t$ at the end, it would never choose to take less, so this does not negatively affect its earnings.

Consider an algorithm $ALG'$ for the original instance that ignores all values less than $w_t$ and follows the decisions of $ALG$ exactly when possible. The only situation in which $ALG'$ performs differently than $ALG$ is when $ALG$ selects $w_t$ after observing all of $U'$, but such a value is not available to $ALG$. By Lemma 22, this happens with probability at most $\epsilon$, implying $ALG' \geq ALG - \epsilon \cdot w_t$. We know that $ALG \geq w_t$, implying the desired inequality $OPT \geq ALG' \geq (1-\epsilon) \cdot ALG$. □

### 4.1.3 PTAS for the More General Case

To move beyond the single-item case into general constant-rank matroids, we begin looking at concentration on flats. If $G(w_j)$ is "sufficiently concentrated" on a flat $F$, then, similarly to the single-item case, we may assume with small error that a basis of $F$ with weights $w_j$ is always "free to select". This allows us to ignore items with weights less than or equal to $w_j$ in $F$.

Now for each value $w_j$, we further partition the elements from the group $G(w_j)$ into three groups $G_{\text{grant}}(w_j)$, $G_{\text{luck}}(w_j)$ and $G_{\text{ignore}}(w_j)$. Let us provide the intuition behind these groups. For the elements in $G_{\text{luck}}(w_j)$, our status tracks the information of whether they have already arrived or not. In some sense, we are not taking any of the elements in $G_{\text{luck}}(w_j)$ for "granted"; so we are ready to keep information about these elements. The group $G_{\text{grant}}(w_j)$ can be seen as the elements that we take for "granted". More precisely, we rely on a large set of elements from $G_{\text{grant}}(w_j)$ to take on value equal to $w_j$; and to arrive sufficiently late, in particular they arrive after all elements in $G_{\text{luck}}(w_t)$, $t = 1, \ldots, j-1$;

and we rely on this large set to also contain independent sets in $\mathcal{M}$ of certain structure. We ignore all elements from $G(w_j)$ that are in neither $G_{\text{grant}}(w_j)$ nor $G_{\text{luck}}(w_j)$, and put them in $G_{\text{ignore}}(w_j)$.

Let $U' = \bigcup_{j=1}^{k} G_{\text{luck}}(w_j)$. Consider the following game. All elements of $U'$ arrive in uniformly random order as in the prophet secretary instance. The values of elements in $U'$ are drawn randomly from their distributions, and the gambler can select any item when it arrives subject to maintaining feasibility, winning its drawn value. After all elements of $U'$ have arrived, the gambler may select any elements of $\bigcup_{j=1}^{k} G_{\text{grant}}(w_j)$ they wish, subject to maintaining feasibility in the matroid. For each element of $G_{\text{grant}}(w_j)$ the gambler selects, they gain a value of $w_j$. In other words, for each $j$, all members of $G_{\text{grant}}(w_j)$ are given deterministic value $w_j$ and have their arrivals moved to after $U'$. Note that no elements from $\bigcup_{j=1}^{k} G_{\text{ignore}}(w_j)$ appear in this instance.

If $|U'|$ is a constant, the expected gain of the optimal online algorithm in the above game can be calculated in polynomial time. This can be done with a simple dynamic programming algorithm in which the state consists of all currently selected elements along with the subset of elements from $U'$ left to arrive. Let $\mathcal{A}$ be such an algorithm that takes in $(\mathcal{M}, \{G_{\text{grant}}(w_j), G_{\text{luck}}(w_j)\}_j)$ and outputs the expected gain of the optimal online algorithm in this game.

Let $\alpha \cdot \mathcal{M}$ denote the matroid over the same ground set $U$, where a set $S \subseteq U$ is independent in $\alpha \cdot \mathcal{M}$ if and only if it can be partitioned into $S = S_1 \cup S_2 \cup \ldots \cup S_\alpha$ where each $S_i$ is independent in $\mathcal{M}$. We use the following algorithm to estimate the expected philosopher's gain, where $r$ represents the rank of $\mathcal{M}$ and $\alpha_j \cdot \mathcal{M}$ is the

**Algorithm 4:** ProphetSecretary$(\mathcal{M}, r, k, p, \epsilon)$

Label $w_j$ and $G(w_j)$ as in items 1 and 2 for $1 \leq j \leq k$;
$\gamma_0 \leftarrow 0, \alpha_0 \leftarrow 0$;
**for** $j = 1, 2, \ldots, k$ **do**
    $\gamma_j \leftarrow \gamma_{j-1} + \alpha_{j-1}$;
    $\alpha_j \leftarrow \lceil \log_{1-(p/(\gamma_j+1))^r}(\epsilon/k) \rceil$;
    $\mathcal{M}_j \leftarrow \alpha_j \cdot \mathcal{M}$;
    Let $F_j$ be a maximal flat of $\mathcal{M}_j$ such that $\mathcal{M}_j|_{F_j}$ admits a basis that contains elements only from $G(w_j)$;
    $G_{\text{grant}}(w_j) \leftarrow F_j \cap G(w_j)$;
    $G_{\text{luck}}(w_j) \leftarrow G(w_j) \setminus G_{\text{grant}}(w_j)$;
**end**
Return $\mathcal{A}(\mathcal{M}, \bigcup_{j=1}^{k} G_{\text{grant}}(w_j), \bigcup_{j=1}^{k} G_{\text{luck}}(w_j))$;

As in Algorithm 3, note that $\alpha_j$ relies only on $j$, $p$, $r$, and $\epsilon$. Therefore, when $k$, $p$, $r$, and $\epsilon$ are constant, all $\alpha_j$ are bounded by some constant. We wish to show that in this case, we also have that $G_{\text{luck}}(w_j)$ is bounded by a constant for each $j$, so that the call to $\mathcal{A}$ runs in polynomial time.

**Lemma 23.** *For every $j = 1, 2, \ldots, k$, we have that $|G_{luck}(w_j)| \leq \alpha_j \cdot r$.*

*Proof.* The statement follows from the definition of $F_j$ in Algorithm 4. By maximality of $F_j$, we have that $\mathcal{M}'_j := \mathcal{M}_j/F_j$ has no non-trivial flat $F'$ such that $\mathcal{M}'_j|_{F'}$ admits a basis that contains elements only from $G(w_j)$. That implies that the set $G_{\text{luck}}(w_j)$ is independent in the matroid $\mathcal{M}_j/F_j$ and so $|G_{\text{luck}}(w_j)| \leq \alpha_j \cdot rank_{\mathcal{M}}(G_{\text{luck}}(w_j)) \leq \alpha_j \cdot r$. □

Therefore, when $k$, $p$, $r$, and $\epsilon$ are constant, we have that Algorithm 4 runs in polynomial time. It remains to bound the error of Algorithm 4.

**Lemma 24.** *For every $j = 1, 2, \ldots, k$, the probability there exists a $B_j \subseteq G_{grant}(w_j)$ such that*

1. *all elements from $B_j$ take on value $w_j$,*
2. *all elements from $B_j$ arrive after the arrival of all elements from $\bigcup_{t=1}^{j-1} G_{luck}(w_t)$, and*
3. *$B_j$ is a basis of $\mathcal{M}|_{F_j}$*

*is at least $1 - \epsilon/k$.*

*Proof.* Consider any $B_j \subseteq G_{\text{grant}}(w_j)$ such that $B_j$ is a basis of $\mathcal{M}|_{F_j}$. The probability that every element of $B_j$ takes on value $w_j$ is at least $p^{|B_j|} \geq p^r$. Further, the probability that all elements of $B_j$ arrive after the arrival of all elements from $\bigcup_{t=1}^{j-1} G_{\text{luck}}(w_t)$ is at least $\left(\frac{1}{|\bigcup_{t=1}^{j-1} G_{\text{luck}}(w_t)|+1}\right)^{|B_j|} \geq \left(\frac{1}{\gamma_j+1}\right)^r$. Therefore, the probability that $B_j$ satisfies the conditions of the lemma is at least $(\frac{p}{\gamma_j+1})^r$, as the arrival order is independent of values drawn.

Due to how we chose $G_{\text{grant}}(w_j)$, there exists $\alpha_j$ disjoint $B_j^{(1)}, B_j^{(2)}, \ldots, B_j^{(\alpha_j)} \subseteq G_{\text{grant}}(w_j)$ bases of $\mathcal{M}|_{F_j}$. Because they are disjoint, the probability of any $B_j^{(i)}$ fulfilling the conditions of the lemma is independent of all other chosen bases. By independence, the probability that none of the $B_j^{(i)}$ fulfill the conditions of the lemma is at most

$$\left(1 - \left(\frac{p}{\gamma_j + 1}\right)^r\right)^{\alpha_j} = \epsilon/k.$$

□

Fixing an instance of prophet secretary, let $ALG$ refer to the output of Algorithm 4, $OPT$ refer to the expected online optimal gain, and $\{F_j, G_{\text{luck}}(w_j), G_{\text{grant}}(w_j)\}$ be as computed in Algorithm 4.

**Lemma 25.** *For an instance $(\mathcal{M}, r, k, p)$ of high-probability constant-rank matroid prophet secretary, let $OPT$ denote the expected philosopher's gain, and $ALG$ be the result of running Algorithm 4 on the instance. Then we have that $OPT \in [(1-\epsilon) \cdot ALG, ALG]$.*

*Proof.* Notice that the game played by $ALG$ has at least the same options as the original game played by $OPT$. This is because we produced the new game by guaranteeing certain elements take on their highest possible value and moving them to the end. So we must have that $ALG \geq OPT$.

Consider an algorithm $ALG'$ for the original game, as played by $OPT$. We define $ALG'$ to follow the decisions of $ALG$ on $\bigcup_{j=1}^{k} G_{\text{luck}}(w_j)$ while ignoring all other elements that arrive before the last element of $\bigcup_{j=1}^{k} G_{\text{luck}}(w_j)$. After $ALG$ has observed all elements of $\bigcup_{j=1}^{k} G_{\text{luck}}(w_j)$, it now has all information about the remaining elements $\bigcup_{j=1}^{k} G_{\text{grant}}(w_j)$ and will immediately select some set $B \subseteq \bigcup_{j=1}^{k} G_{\text{grant}}(w_j)$.

In particular, due to the greedy selection property of matroids, $ALG$ will greedily select items from $G_{\text{grant}}(w_1)$ until the rest are infeasible, then proceed to do the same from $G_{\text{grant}}(w_2)$ and so on. Due to Lemma 24, we have with probability at least $1 - \frac{\epsilon}{k}$, $ALG'$ will be able to select items from $G_{\text{grant}}(w_j)$ to model the items selected by $ALG$. By the union bound, the probability that $\mathcal{A}$ can do this for all $j$ is at least $1 - \epsilon$. This implies $OPT \geq ALG' \geq (1-\epsilon) \cdot ALG$, as wanted. □

Lemma 25 and the fact that Algorithm 4 runs in polynomial-time implies that Algorithm 4 is a PTAS and thus Theorem 9 holds.

# Chapter 5

# Computing the Prophet's Revenue

It is natural to wonder whether the 2-competitive algorithm for matroid prophet inequalities from [KW12] can be made to run in polynomial time, as it would immediately give a 2-approximation to the profit of the philosopher. The crux of that algorithm involves calculating the expected maximum-weight basis of a matroid. In other words, one needs to calculate the expected gain of the prophet. This in itself would be sufficient to attain a 2-approximation for Matroid Bayesian Online Selection, as [KW12] shows the optimal philosopher gain is at least half of the prophet's gain.

In this chapter, we show that it is #P-hard to calculate the expected prophet gain on planar graphic matroids in polynomial time via a reduction from network reliability. On the positive side, we give an FPRAS for estimating the expected weight of the maximum-weight basis of a matroid - that is, estimating the prophet's expected gain. The algorithm for this is based on sampling and relatively straightforward. As far as we know, this result has not been mentioned in the literature before. However, we expect that it is already known by some due to its simplicity. Finally, we show that one can compute the expected prophet gain exactly in the special case of laminar matroids.

## 5.1 Hardness in Planar Graphic Matroids

**Theorem 11.** *It is #P-hard to compute the expected prophet gain, even in planar graphic matroids. Further, it is #P-hard to determine how to* act *in* `GBOS` *on planar graphic matroids.*

Note the distinction: we can think of the prophet as the gambler in a specific case of matroid Bayesian online selection in which elements are ordered by decreasing value. In this case, the optimal strategy is trivial - just accept anything with non-zero value. We show, despite this, that it is #P-hard to compute the expected gain of this strategy. This immediately implies that GBOS is hard on planar graphic matroids, but it does not show that finding the optimal strategy is #P-hard.

To show Theorem 11, we reduce from the two-terminal network reliability problem.

**Definition 6.** *In the* **two-terminal network reliability problem**, *the input is a graph $G = (V, E)$, two special terminals $s, t \in V$, and a probability $p_e$ associated with each edge $e \in E$. Each edge $e \in E$ is present with independent probability $p_e$. The answer to the problem is the probability that $s$ and $t$ are in the same connected component.*

In [Pro86], it is shown that two-terminal network reliability is #P-hard, even in planar graphs where both terminals lie on the same face. We reduce from network reliability to GBOS.

Let $(G = (V, E), s, t, \{p_e\}_{e \in E})$ be the input to a planar two-terminal network reliability instance as in [Pro86]. Define $G' = (V, E \cup \{st\})$.

**Lemma 26.** *Let $MWB(G)$ indicate the expected maximum weight basis of the graphic matroid associated with $G$ where each edge attains value 1 with probability $p_e$ and value 0 otherwise. Let $MWB(G')$ indicate the expected maximum weight basis of the graphic matroid associated with $G'$ where $st$ always has value 1, and all other edges have value 1 with probability $p_e$ and value 0 otherwise.*

*Then $MWB(G') - MWB(G)$ is equal to $1-p$, where $p$ is the answer to the two-terminal network reliability instance.*

*Proof.* All edges have equal value when non-zero, so it is optimal for the prophet to consider edges in any order and select any that have non-zero value due to the greedy selection property of matroids. Fix some order of $E$ and consider the prophet algorithm which observes the edges in this order in $G$, and in $G'$ observes the edges in this order followed by the edge $st$. By coupling the realization of non-$st$ edges in $G$ and $G'$, we can assume that the two algorithms will act identically on all edges in $E$. Therefore, the only situation in which their values will differ is when $s$ and $t$ are not connected in $G$. In this case, the prophet for $G'$ wins one extra value.

So the expected difference between the maximum weight basis in these two graphs is exactly the probability that $s$ and $t$ lie in different components, that is, $1 - p$. □

We note that $G'$ is still a planar graph, as the reduction from [Pro86] has $s$ and $t$ in the same face of the graph. Because two-terminal planar network reliability is #P-hard, this implies that calculating the prophet's expected gain in planar graphs is #P-hard.

While it is #P-hard to estimate the expected maximum weight basis of these graphs, the decisions to make in the produced `GBOS` instance are trivial. Because all values are either 1 or 0, it is optimal for the philosopher to select every edge with value 1 it sees and reject every edge with value 0. To show it is #P-hard to determine how the philosopher will *act*, consider the following example.

Let $G$ and $G'$ be as above. We will modify only the values and arrival order of edges in $G'$.

- For each edge $e \in E$, let the value of $e$ be 1 with probability $p_e$ and 0 otherwise.
- Let the value of $st$ be $p \in [0, 1]$ with probability 1.
- Finally, order the edges so that $st$ is first and the rest appear in any order.

The only interesting choice for the gambler is whether to take $st$ - after this, they should just take all elements with value 1 and discard all elements with value 0. To maximize their expected value, the gambler should select $st$ if they believe that $p$ is greater than the probability that $s$ and $t$ will be connected by the rest of the edges, and discard $st$ otherwise. Therefore, it is #P-hard for the gambler to determine how to act in this case.

## 5.2 FPRAS for all Matroids

In this section, we show that despite the hardness of computing the expected gain of the prophet, there is a fully polynomial-time randomized approximation scheme. Thus, we can approximate the expected gain of the prophet arbitrarily well.

**Theorem 12.** *There is an FPRAS for calculating the expected gain of the prophet in all matroids.*

*Proof.* Consider a matroid $\mathcal{M}$ over a ground set $U$, and let $u \in U$ be any element. Suppose that the possible non-zero values $u$ could possibly take are $w_1 > w_2 > \ldots > w_k$. Let $q_i$ indicate the probability that $u$ takes on value $w_i$ for $1 \leq i \leq k$. Then we can define a new matroid $\mathcal{M}'$ where $u$ is replaced with $k$ parallel elements $e_1, e_2, \ldots, e_k$, where $e_i$ takes

on value $v_i$ with probability $q_i/(1-\sum_{j=1}^{i-1} q_j)$ and value 0 otherwise. Then the expected maximum weight basis of $\mathcal{M}'$ is equal to the expected maximum weight basis of $\mathcal{M}$. This is because the distribution over the maximum value of $e_1, e_2, \ldots, e_k$ is identical to the distribution over the values of $u$. Note that each $e_i$ is equipped with a weighted Bernoulli distribution. Thus, by repeating this process for all $u \in U$, we can assume without loss of generality that all elements of $U$ are equipped with a weighted Bernoulli distribution.

Given this assumption of weighed Bernoulli distributions, let $p_i$ and $v_i$ indicate the probability and value of the $i$th element of $\mathcal{M}$ - this element takes on value $v_i$ with probability $p_i$ and value 0 otherwise. We can now state our algorithm, where $cl(S)$ indicates the closure of $S$ — the set of all $u \in U$ such that the rank of $S$ equals the rank of $S \cup \{u\}$.

**Algorithm 5:** ProphetFPRAS($\mathcal{M} = (U, \mathcal{I})$, $\epsilon$)

```
Sort the elements of U = {u_1, u_2, ..., u_n} by non-increasing order of value v_i ;
t ← 64n^5 / ε^4 ;
for i = 1, ..., n do
    cnt ← 0 ;
    repeat t times
        Sample S ⊆ {u_1, u_2, ..., u_{i-1}} by including u_j with probability p_j ;
        if u_i ∉ cl(S) then
            cnt ← cnt + 1 ;
        end
    end
    X_i ← cnt / t ;
end
A = Σ_{i=1}^{n} X_i · p_i · v_i ;
Return A ;
```

The intuition behind Algorithm 5 is that, due to the greedy selection property of matroids, it is optimal for the prophet to sort the elements in non-increasing order of value and greedily select elements with non-zero values. So the prophet's gain is $OPT = \sum_{i=1}^{n} q_i \cdot p_i \cdot v_i$, where $q_i$ is the probability that $u_i$ is not in the closure of previously selected elements. In our algorithm, we estimate $q_i$ with $X_i$ via sampling.

Call an element $u_i$ "big" if $q_i \geq \frac{\epsilon}{4n^2}$, and "small" otherwise. We will bound the error on $X_i$ separately for big and small elements. It is clear to see that for all $X_i$, $\mathbb{E}[X_i] = q_i$ and $\mathbf{Var}[X_i] \leq \frac{1}{t}$. Therefore, for big elements $u_i \in U$, we have by the second moment method

that

$$\begin{aligned}\mathbf{Pr}[|X_i - q_i| \geq \epsilon \cdot q_i] &\leq \frac{\mathbf{Var}[X_i]}{\epsilon^2 \cdot q_i^2} \\ &\leq \frac{\epsilon^4}{64n^5 \cdot \epsilon^2 \cdot q_i^2} \\ &\leq \frac{\epsilon^4 \cdot 16n^4}{64n^5 \cdot \epsilon^2 \cdot \epsilon^2} \\ &= \frac{1}{4n}\end{aligned}$$

For small elements $u_i \in U$, we use non-negativity and Markov's inequality to find that

$$0 \leq X_i \leq 4n \cdot q_i$$

with probability at least $1 - \frac{1}{4n}$. By union bound, these inequalities hold for all elements with probability at least $\frac{3}{4}$. We show that these bounds imply that $\mathcal{A}$ approximates $OPT$ well.

The rest of the argument hinges on small elements contributing almost nothing to $OPT$. We know that for any element $u_i \in U$, it must be the case that $p_i \cdot v_i \leq OPT$, or else the gambler could out-perform $OPT$ by simply waiting for and selecting $u_i$. So we have that $q_i \cdot p_i \cdot v_i \leq \frac{\epsilon}{4n^2} \cdot OPT$ for any small element $u_i \in U$. Even if all elements are small, their total contribution is still at most $\frac{\epsilon}{4n} \cdot OPT$.

Because we know that $0 \leq X_i \leq 4n \cdot q_i$ with good probability for small elements, the total error incurred from overestimating the probability of selecting small elements will be at most $\epsilon \cdot OPT$ with good probability. For large elements, we only miss by a factor of $\epsilon$ with good probability, so they contribute an error of at most $\epsilon \cdot OPT$. So we find that

$$(1 - 2\epsilon) \cdot OPT \leq \mathcal{A} \leq (1 + 2\epsilon) \cdot OPT$$

with probability at least $\frac{3}{4}$, as wanted.

□

## 5.3 Exact Algorithm for Laminar Matroids

The separation between graphic and laminar matroids observed in Chapters 2 and 3 continues to occur when calculating the prophet's gain, as we show in this section that the prophet's expected gain can be computed exactly in laminar matroids.

For simplicity, assume that

- there is a unique maximal bin in the matroid,
- there are no empty bins,
- all elements are contained in some "leaf" bin that only contains this element, and
- all non-leaf bins have exactly two maximal bins contained within them.

Note that these conditions can be assumed without loss of generality, as we can always either introduce "dummy" bins with infinite capacity or remove redundant bins without affecting independence of sets.

To obtain our algorithm, we give a dynamic programming algorithm that, given a bin $B \in \mathcal{L}$ and integer $k \in [0, c(B)]$, calculates the probability that the rank of $\mathcal{M}|_B$ is equal to $k$.

Correctness of Algorithm 6 can be shown by inducting on the size of bins $B$. Note that the rank $r(B)$ of a non-leaf bin $B$ with maximal contained small bins $B_1$ and $B_2$ is equal to $\max\{c(B), r(B_1) + r(B_2)\}$. Thus, we need to handle the special case of $k = c(B)$ separately, as this case doesn't require $r(B_1) + r(B_2)$ to equal $k$.

To achieve polynomial runtime, simply add a memo-ization step to turn Algorithm 6 into a dynamic programming algorithm. Then the number of states is bounded by $|\mathcal{L}| \cdot n$, and each state takes at most $O(n)$ computation time, giving a polynomial time algorithm.

The expected rank of a matroid with unique maximal bin $B$ can then be computed as

$$\sum_{k=0}^{n} k \cdot dp(\mathcal{M}, B, k).$$

Now, making the same assumption as in Algorithm 5, that each element's distribution is a weighted Bernoulli achieving value $v_i$ with probability $p_i$, sort the elements $u_1, u_2, \ldots, u_n$ in order of non-increasing $v_i$.

Let $\mathcal{M}_i := M|_{\{u_1, u_2, \ldots, u_i\}}$. Then, by the greedy selection property of matroids, the probability $q_i$ that item $u_i$ is contained in the maximum weight basis of $\mathcal{M}$ is the expected rank of $\mathcal{M}_i$ minus the expected rank of $\mathcal{M}_{i-1}$. Thus, the expected contribution of $u_i$ is $q_i \cdot v_i$. Adding these probabilities for all elements produces the expected weight of the maximum weight basis of $\mathcal{M}$.

**Algorithm 6:** dp($\mathcal{M}$, $B$, $k$)

```
if B has no bins contained within it then
    Let u_i denote the element contained in B ;
    if k = 0 then
        Return 1 − p_i ;
    end
    if k = 1 then
        Return p_i ;
    end
    Return 0 ;
end
if k > c(B) then
    Return 0 ;
end
if k = c(B) then
    out ← 1 ;
    for i = 0, 1, ..., k − 1 do
        out ← out − dp(M, B, i) ;
    end
    Return out ;
end
Let B_1, B_2 be the maximal small bins contained in B ;
out ← 0 ;
for i = 0, 1, ..., k do
    out ← out + dp(M, B_1, i) · dp(M, B_2, k − i) ;
end
Return out ;
```

# References


[ACK18] Yossi Azar, Ashish Chiplunkar, and Haim Kaplan. Prophet secretary: Surpassing the 1-1/e barrier. In *Proceedings of the 2018 ACM Conference on Economics and Computation*, pages 303–318, 2018.

[AEE+17] Melika Abolhassani, Soheil Ehsani, Hossein Esfandiari, MohammadTaghi Hajiaghayi, Robert Kleinberg, and Brendan Lucier. Beating 1-1/e for ordered prophets. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 61–71, 2017.

[AHL12] Saeed Alaei, MohammadTaghi Hajiaghayi, and Vahid Liaghat. Online prophet-inequality matching with applications to ad allocation. In *Proceedings of the 13th ACM Conference on Electronic Commerce*, pages 18–35, 2012.

[AKW14] Pablo D Azar, Robert Kleinberg, and S Matthew Weinberg. Prophet inequalities with limited information. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 1358–1377. SIAM, 2014.

[AKW19] Pablo D Azar, Robert Kleinberg, and S Matthew Weinberg. Prior independent mechanisms via prophet inequalities with limited information. *Games and Economic Behavior*, 118:511–532, 2019.

[Ala11] Saeed Alaei. Bayesian combinatorial auctions: Expanding single buyer mechanisms to many buyers. In *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, pages 512–521, 2011.

[ANSS19] Nima Anari, Rad Niazadeh, Amin Saberi, and Ali Shameli. Nearly optimal pricing algorithms for production constrained and laminar bayesian selection. In *Proceedings of the 2019 ACM Conference on Economics and Computation*, page 91–92, 2019.

[ASZ20] Shipra Agrawal, Jay Sethuraman, and Xingyu Zhang. On optimal ordering in the optimal stopping problem. In *Proceedings of the 21st ACM Conference on Economics and Computation*, EC '20, page 187–188, New York, NY, USA, 2020. Association for Computing Machinery.

[BDML22] Mark Braverman, Mahsa Derakhshan, and Antonio Molina Lovett. Max-weight online stochastic matching: Improved approximations against the online benchmark. In *Proceedings of the 23rd ACM Conference on Economics and Computation*, EC '22, page 967–985, New York, NY, USA, 2022. Association for Computing Machinery.

[BDP+24] Mark Braverman, Mahsa Derakhshan, Tristan Pollner, Amin Saberi, and David Wajc. New philosopher inequalities for online bayesian matching, via pivotal sampling. *arXiv preprint arXiv:2407.15285*, 2024.

[BHK+24] Kiarash Banihashem, MohammadTaghi Hajiaghayi, Dariusz R Kowalski, Piotr Krysta, and Jan Olkowski. Power of posted-price mechanisms for prophet inequalities. In *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 4580–4604. SIAM, 2024.

[CFH+17] José Correa, Patricio Foncea, Ruben Hoeksma, Tim Oosterwijk, and Tjark Vredeveld. Posted price mechanisms for a random stream of customers. In *Proceedings of the 2017 ACM Conference on Economics and Computation*, pages 169–186, 2017.

[CGKM20] Shuchi Chawla, Kira Goldner, Anna R Karlin, and J Benjamin Miller. Non-adaptive matroid prophet inequalities. *arXiv preprint arXiv:2011.09406*, 2020.

[CST24] Liyan Chen, Nuozhou Sun, and Zhihao Gavin Tang. Setting targets is all you need: Improved order competitive ratio for online selection. *arXiv preprint arXiv:2406.15192*, 2024.

[CSZ21] Jose Correa, Raimundo Saona, and Bruno Ziliotto. Prophet secretary through blind strategies. *Mathematical Programming*, 190(1-2):483–521, 2021.

[DGR+23] Paul Dütting, Evangelia Gergatsouli, Rojin Rezvan, Yifeng Teng, and Alexandros Tsigonias-Dimitriadis. Prophet secretary against the online optimal. In *Proceedings of the 24th ACM Conference on Economics and Computation*, EC '23, page 561–581, New York, NY, USA, 2023. Association for Computing Machinery.

[DP24] Ian DeHaan and Kanstantsin Pashkovich. Matroid bayesian online selection. *arXiv preprint arXiv:2406.00224*, 2024.

[EFGT20] Tomer Ezra, Michal Feldman, Nick Gravin, and Zhihao Gavin Tang. Online stochastic max-weight matching: prophet inequality for vertex and edge arrival models. In *Proceedings of the 21st ACM Conference on Economics and Computation*, pages 769–787, 2020.

[EHKS18] Soheil Ehsani, MohammadTaghi Hajiaghayi, Thomas Kesselheim, and Sahil Singla. Prophet secretary for combinatorial auctions and matroids. In *Proceedings of the twenty-ninth annual acm-siam symposium on discrete algorithms*, pages 700–714. SIAM, 2018.

[EHLM17] Hossein Esfandiari, MohammadTaghi Hajiaghayi, Vahid Liaghat, and Morteza Monemizadeh. Prophet secretary. *SIAM Journal on Discrete Mathematics*, 31(3):1685–1701, 2017.

[FJ81] Greg N Frederickson and Joseph Ja'Ja'. Approximation algorithms for several graph augmentation problems. *SIAM Journal on Computing*, 10(2):270–283, 1981.

[FLX18] H Fu, J Li, and P Xu. A ptas for a class of stochastic dynamic programs. chatzigiannakis i, kaklamanis c, marx d, sannella d, eds. In *Proc. 45th Internat. Colloquium on Automata, Languages, and Programming.(ICALP 2018), Prague, Czech Republic*, pages 1–56, 2018.

[FNS21] Yiding Feng, Rad Niazadeh, and Amin Saberi. Two-stage stochastic matching with application to ride hailing. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2862–2877. SIAM, 2021.

[FNS22] Yiding Feng, Rad Niazadeh, and Amin Saberi. Near-optimal bayesian online assortment of reusable resources. In *Proceedings of the 23rd ACM Conference on Economics and Computation*, pages 964–965, 2022.

[FSZ16] Moran Feldman, Ola Svensson, and Rico Zenklusen. Online contention resolution schemes. In *Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete algorithms*, pages 1014–1033. SIAM, 2016.

[GJS74] Michael R Garey, David S Johnson, and Larry Stockmeyer. Some simplified np-complete problems. In *Proceedings of the sixth annual ACM symposium on Theory of computing*, pages 47–63, 1974.

[Gup17] Anupam Gupta. Lecture Notes, IPCO Summer School, 2017.

[GW19] Nikolai Gravin and Hongao Wang. Prophet inequality for bipartite matching: Merits of being simple and non adaptive. In *Proceedings of the 2019 ACM Conference on Economics and Computation*, pages 93–109, 2019.

[HK82] Theodore P Hill and Robert P Kertz. Comparisons of stop rule and supremum expectations of i.i.d. random variables. *The Annals of Probability*, 10(2):336–345, 1982.

[HKS07] Mohammad Taghi Hajiaghayi, Robert Kleinberg, and Tuomas Sandholm. Automated online mechanism design and prophet inequalities. In *Proceedings of the 22nd National Conference on Artificial Intelligence - Volume 1*, AAAI'07, page 58–65. AAAI Press, 2007.

[KKL04] Guy Kortsarz, Robert Krauthgamer, and James R Lee. Hardness of approximation for vertex-connectivity network design problems. *SIAM Journal on Computing*, 33(3):704–720, 2004.

[KS77] Ulrich Krengel and Louis Sucheston. Semiamarts and finite values. *Bulletin of the American Mathematical Society*, 83(4):745–747, 1977.

[KW12] Robert Kleinberg and Seth Matthew Weinberg. Matroid prophet inequalities. In *Proceedings of the Forty-Fourth Annual ACM Symposium on Theory of Computing*, page 123–136, 2012.

[Luc17] Brendan Lucier. An economic view of prophet inequalities. *SIGecom Exch.*, 16(1):24–47, sep 2017.

[MR02] Michael Molloy and Bruce Reed. *Graph colouring and the probabilistic method*, volume 23. Springer Science & Business Media, 2002.

[NSW23] Joseph Naor, Aravind Srinivasan, and David Wajc. Online dependent rounding schemes, 2023.

[Oxl06] James G. Oxley. *Matroid Theory*. Oxford graduate texts in mathematics. Oxford University Press, 2006.

[PPSW21] Christos Papadimitriou, Tristan Pollner, Amin Saberi, and David Wajc. Online stochastic max-weight bipartite matching: Beyond prophet inequalities. In *Proceedings of the 22nd ACM Conference on Economics and Computation*,

EC '21, page 763–764, New York, NY, USA, 2021. Association for Computing Machinery.

[Pro86] J. Scott Provan. The complexity of reliability computations in planar and acyclic graphs. *SIAM Journal on Computing*, 15(3):694–702, 1986.

[PS23] Kanstantsin Pashkovich and Alice Sayutina. Non-adaptive matroid prophet inequalities. *arXiv preprint arXiv:2301.01700*, 2023.

[Rou21] Tim Roughgarden, editor. *Beyond Competitive Analysis*, page 529–546. Cambridge University Press, 2021.

[RS17] Aviad Rubinstein and Sahil Singla. Combinatorial prophet inequalities. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1671–1687. SIAM, 2017.

[Rub16] Aviad Rubinstein. Beyond matroids: Secretary problem and prophet inequality with general constraints. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 324–332, 2016.

[RWW19] Aviad Rubinstein, Jack Z Wang, and S Matthew Weinberg. Optimal single-choice prophet inequalities from samples. *arXiv preprint arXiv:1911.07945*, 2019.

[SC84] Ester Samuel-Cahn. Comparison of threshold stop rules and maximum for independent nonnegative random variables. *the Annals of Probability*, pages 1213–1216, 1984.

[SS21] Danny Segev and Sahil Singla. Efficient approximation schemes for stochastic probing and prophet problems. In *Proceedings of the 22nd ACM Conference on Economics and Computation*, pages 793–794, 2021.

[Waj23] David Wajc. Personal Communication, 2023.