Essays on Discrete Optimization:

Optimal Stopping and Popular Matchings

Xingyu Zhang

Submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy
under the Executive Committee
of the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2022

Abstract

Essays on Discrete Optimization:

Optimal Stopping and Popular Matchings

Xingyu Zhang

This thesis studies two discrete optimization problems: ordering problems in optimal stopping theory and popular matchings. The main goal of this thesis is to find the boundary between NP-hardness and tractability for these problems, and whenever possible, designs polynomial-time algorithms for the easy cases and approximation schemes or prophet inequalities for the hard cases.

In the first part of the thesis, we study ordering problems in optimal stopping theory. In the optimal stopping problem, a player is presented with n random variables $X_1, \ldots, X_n$, whose distributions are known to the player, but not their realizations. After observing the realization of X_i, the player can choose to stop and earn reward X_i, or reject X_i and probe the next variable X_{i+1}. If X_i is rejected, it cannot be accepted in the future. The goal of the player is to maximize the expected reward at stopping time. If the order of observation is fixed, the player can find the optimal stopping criteria using a dynamic program. In this thesis, we investigate the variant in which the player is able to choose the order of observation. What is the best ordering and what benefits does ordering bring?

Chapter 2 introduces the optimal ordering problem in optimal stopping theory. We prove that the problem of finding an optimal ordering is NP-hard even in very restricted cases where the support of each distribution has support on at most three points. Next, we prove an FPTAS for the

hardness case and provide a tractable algorithm and a prophet inequality for two-point distributions. Chapter 3 studies the optimal ordering problem when the player can choose $k > 1$ rewards before stopping. We show that finding an optimal static ordering is NP-hard even for very simple two-point distributions. Next, we prove an FPTAS for the hardness case and give prophet inequalities under static and dynamic policies for two-point distributions.

In the second part of the thesis, we study popular matchings. Suppose we are given a bipartite graph with independent sets A and B. Each vertex in A has a ranked order of preferences on the vertices in B, and vice versa. A matching M is popular if for any other matching M', the number of vertices that prefer M is at least as much as the number of vertices that prefer M'.

Chapter 4 studies popular matchings. In the first part, we provide a general reduction which, through minor adjustments, proves NP-Hardness for a variety of different questions, including that of finding a max-weight popular matching. In the second part, we restrict our attention to graphs of bounded treewidth and provide a tractable algorithm for finding a max-weight popular matching.

Table of Contents

Acknowledgements

My Ph.D. journey has had many ups and downs, sometimes disappointed and frustrated, but many times also joyful from overcoming the challenges along the way. Looking back, I've definitely grown significantly as a researcher and gained increased self-confidence in my own ability to solve problems.

I can't thank my advisors Jay and Shipra enough for their continued patience and guidance throughout my studies. I remember, as a naïve senior in college, I sent out a few cold emails to a bunch of faculty at Columbia for possible research over the summer, and am eternally grateful that Jay replied to that email. Their passion for research and teaching has been a continuous source of motivation for me. I learned so much meeting with them and seeing how they approach problems. Thank you so much Jay and Shipra for all your kindness.

In addition, I want to give a huge thanks to Yuri Faenza. I met Yuri during my second year and he has been instrumental in mentoring me during this time and has been a wonderful person to get to know. Working on popular matchings with Yuri and discussing and brainstorming with him lit a spark in me - research was exciting and reminded me ogmf how much I liked it. I really looked forward to working with Yuri in our weekly meetings.

I'm grateful for all of my peers that I've met here at Columbia. Especially Goutam Kumar, Apurv Shukla, and Li Fengpei, my time here would have been entirely different without you all.

I want to thank my parents. They are my original role models and without the scholarly and nurturing environment I grew up in, I would never have contemplated doing a Ph.D. in the first place.

Dedication

To my parents and sister for everything.

Chapter 1: Introduction

Discrete optimization problems are foundational to operations research and theoretical computer science and these problems have wide ranging applications in industry, such as supply chains, auction design, scheduling, and resource allocation. Some examples of its use include packing items onto a cargo truck, matching drivers on a ride-sharing service to passengers, and assigning jobs to machines to minimize the overall completion time. In many practical settings, the scale of the problem is very large, and thus a main concern is to find a good solution relatively quickly.

The high-level goal of discrete optimization is to find the best solution from a discrete set of candidate solutions. The main challenge is that this set of all possible solutions can be huge and can grow exponentially with the size of the input. Therefore, although it is possible to use a brute force technique to test every solution to find the best one, it is impossible to do so in practice. For example, in the traveling salesman problem, one could try every possible route, but for n cities, this amounts to $n!$ possible solutions, which is impossible for a computer to solve except for small values of n. Therefore, we are more concerned with the best solution we can get that can be found quickly. In order to do this, we ask ourselves two questions:

1. Is there a fast (polynomial-time) algorithm to find an optimal solution?

2. Is there a fast (polynomial-time) algorithm to find an approximate solution?

To answer the first question, we try to determine whether a problem is NP-hard. A negative answer to this question means that there is a fast, polynomial-time algorithm that will return the optimal solution. There is no easy way to determine this and sometimes we could have two very similar problems where one is NP-hard and the other is not. For example, a shortest path problem is easily solved using Dijkstra's algorithm, but the related longest path problem is NP-hard. Many

times, proving a problem is NP-hard requires a construction that is tailor made to the problem at hand.

When a problem is NP-hard, our second question to ask is whether there is a fast, polynomial-time algorithm to approximate the optimal solution. Associated with this question is whether this approximation that we find could be further improved. That is, what is the best approximation that we can find for this problem that is polynomial-time?

The goal of this thesis is to examine two case studies in discrete optimization and ask ourselves: What is the best possible solution that can be found quickly? The first case study is on optimal stopping theory and prophet inequalities and the second case study is on popular matchings. Throughout this dissertation, we focus on answering the two questions above for these two case studies. A general approach we have is to find the simplest instance for which the given problem is NP-hard. Then, we show that any simpler problem is in fact tractable, and then we try to find the best possible approximation for the simplest NP-hard case. Answering this set of questions will fully illuminate the boundary between easy and hard instances.

We will now introduce the two case studies.

1.1 Optimal Stopping

1.1.1 Optimal Ordering in the Optimal Stopping Problem

The first topic we study is in the area of optimal stopping and prophet inequalities. The prophet inequality was first introduced by Krengel and Sucheston [43] and considers a setting where there is one player and n random variables $X_1, \ldots, X_n$. The player knows the distributions of the random variables, but does not know their realizations. He then probes the variables one at a time in an adversarial order (the player is aware of the order but cannot choose it). Upon probing X_i, the realization of X_i is revealed to the player, and the player can then decide to either accept or reject X_i. If he chooses to accept X_i, the game ends, but if he rejects X_i, the process continues. The question is, when should the player stop and accept the random variable?

One way to measure the performance of a stopping policy is with the concept of the prophet

inequality. The prophet inequality compares the expected reward of a stopping policy against $E[\max\{X_1, \ldots, X_n\}]$, which is the expected hindsight reward, or the reward of a "prophet" who can see the realizations a priori, and hence can just choose the largest reward. Krengel and Sucheston [43] proved a $1/2$ prophet inequality for this problem, which shows that a player can achieve, on expectation, half of the hindsight reward. Later, Samuel-Cahn [51] showed that this can be achieved using a *threshold* policy, where a suitable threshold θ is defined, and the player picks the first variable whose realized value is larger than θ. In addition, there is a simple example to show that this inequality is tight: let $X_1 = 1$ deterministically and let X_2 equal $1/\epsilon$ with probability ϵ, else 0. For the order (X_1, X_2), the player can only get an expectation of 1, regardless of whether he accepts or rejects X_1, but the prophet's reward is $\epsilon(1/\epsilon) + (1 - \epsilon) = 2 - \epsilon$.

The prophet inequality has a natural connection to posted price mechanisms under welfare maximization, and this was pointed out by Hajiaghayi et al. and Chawla et al. ([32, 9]). Suppose there is one item and n bidders whose private valuations are drawn from X_i. Using a suitable threshold policy (see Samuel-Cahn [51] or Kleinberg and Weinberg [41]), we immediately get that a posted price mechanism returns on expectation at least $1/2$ of the optimal mechanism from Myerson's lemma [46]. We can also apply the prophet inequality to revenue maximization using the standard mapping of expected revenue to expected virtual welfare to get a very simple mechanism for revenue maximization.

Since the original prophet inequality, there have been prophet inequality results for a variety of other cases. A useful old survey by Hill can be found in [34], and some recent surveys are from Lucier [45] and Correa et al. [13]. Much of the work deals with cases where the random variables are presented in either adversarial order, or in uniformly random order (this model is called Prophet Secretary).

In this dissertation, we are mainly concerned with the free order case, where the player has the freedom to choose both the order to examine the variables, as well as the stopping criteria. In many practical situations, the player does have this power. Imagine a scenario where a hiring manager needs to hire a candidate and he has n candidates with prior known distributions $X_1, \ldots, X_n$. Upon

interviewing a candidate X_i, the hiring manager sees X_i's realization and must make a here-and-now decision on whether to hire the candidate or to pass. In this situation, the hiring manager has some freedom to choose who to interview next. In a posted price mechanism, $X_1, \ldots, X_n$ may represent the (hidden) valuations of n bidders. As the seller, you may be able to choose the order of the bidders to approach. These examples show that the ability to order comes up frequently in real life.

The hope with order selection is that the flexibility to choose the order may improve on the expected reward. One way we can compare between policies is through the prophet inequality. In the tight example for the $1/2$ prophet inequality, if we are able to choose the order, then the problem is trivial: Always examine X_2 first and accept if $1/\epsilon$ is realized, otherwise reject and accept X_1. The expectation of the policy actually equals the prophet's reward, so this shows that in some cases there is a gap of 2 between the worst ordering and the best ordering. In the prophet secretary setting where X_i are examined in uniformly random order, the best known approximation is by Correa et al. [11] who prove a 0.669 prophet inequality. Hence, the free order case must be at least 0.669, but we note that it must be less than 0.745, which is the prophet inequality in the IID case (see Correa et al. [12]).

In Chapter 2, we present our results on the free order optimal stopping problem. Of primary importance is understanding the computational complexity of finding the best ordering. Since there are $n!$ possible permutations, enumerating all of the possibilities and using brute force search does not work. We prove that the problem is NP-Hard even in a restricted case where all of the random variables have support on three points $\{0, m_i, 1\}$, with the same left and right endpoints. In addition, we provide a fully polynomial time approximation scheme (FPTAS) for this case. Given the difficulty of the three point case, we next consider the hardest simpler case, which are two-point distributions. We show that there exists a polynomial time algorithm to find the optimal ordering in this setting, thus effectively settling the computational complexity of the free order problem. As a side note, we show that in the case of two-point distributions, there is a 0.8 prophet inequality, and this is tight. This does not improve on the existing upper bound of 0.745. However, since many

upper bounds that have been historically constructed utilize two-point distributions, this prophet inequality shows that you need more complex distributions to improve on the upper bound. The work in this chapter is done jointly with Shipra Agrawal and Jay Sethuraman.

1.1.2 Optimal Ordering in the k-Secretary Problem

Here we consider an extension of the previous optimal ordering problem to the case where the player can choose k random variables where $k > 1$. Again, we have random variables $X_1, \ldots, X_n$ whose distributions are known to the player but not their realizations. The objective is to maximize the sum of all of the rewards chosen by the player. We are concerned with the free order case where the player has the power to choose the stopping policy as well as the order of variables to examine. We can ask many of the same questions that we asked for the previous problem. Which variable should we examine next? What is the computational complexity of finding the optimal ordering? Can we derive any prophet inequalities in this case?

There is one main difference between the $k > 1$ and the $k = 1$ case. In the $k = 1$ case, we can fix an ordering a priori, and this is optimal, meaning that we do not have to change the ordering later. In the $k > 1$ setting, it is usually optimal for a policy to be *dynamic*, meaning that the optimal policy determines the next variable to examine based on all of the realizations up to that point in time. In contrast, a *static* policy fixes an ordering a priori and does not change it online.

Many practical considerations require $k > 1$. For example, in the hiring question, we may want to hire more than one candidate, and in the posted price setting, the seller may have more than one identical item to sell. In addition, as mentioned before, many practical situations allow one to change the ordering; e.g. the hiring manager can choose the order of candidates.

Furthermore, studying the ordering problem is relevant from theoretical observations. The prophet inequality extends to the $k > 1$ case, where it's naturally defined as the sum of the k largest variables. The prophet inequality can then be defined as the expected reward of the player compared to the expected prophet's reward. Hajiaghayi et al. [32] give an upper bound of $1-O(\frac{1}{\sqrt{k}})$ on the prophet inequality under adversarial ordering for large k, and Alaei [2] later shows that

this bound is tight. The example for the upper bound they give considers three sets A, B, C. A consists of variables that are deterministically equal to 1, B consists of variables that are two-point distributions with support on $\{0, 2\}$, and C consists of variables that are deterministically equal to 0. The whole difficulty is solely because the player *must* examine A before B, and hence does not know how many variables in A to accept. If he accepts too many in A, then he potentially loses spots for variables in B that realize 2. If he accepts too few in A, he loses if too many variables in B realize 0. Hence, he needs to balance against these two possibilities. However an optimal static ordering for this problem is trivial: Examine B first, then A, then C. For this case, the optimal static ordering returns the prophet's reward. A natural question to ask is how much does ordering help, even in the case of two-point distributions?

In Chapter 3, we present our results on the free order optimal stopping problem where $k > 1$. We begin by studying the computational complexity of finding the optimal static ordering. We already showed that the $k = 1$ ordering problem is NP-hard in Chapter 2 for three-point distributions, so what is there left to do? Although the previous result extends to the $k > 1$ case, we prove that the problem remains NP-hard even for two-point distributions. Furthermore, the two-point distributions are very simple: all but one of the variables have support on $\{0, 1\}$. Even simpler cases, where all distributions have support on $\{0, 1\}$ are trivial to solve since any ordering is optimal. Because of the complexity of this problem, we focus solely on two-point distributions in this chapter. We next show that there exists an FPTAS for two-point distributions for $k = 2$. We conclude the chapter with prophet inequality results for both static and dynamic policies. For the "static" prophet inequality, we provide an algorithm that computes a static ordering with a prophet inequality of $1 - O(\frac{\log(k)^{1/4}}{k^{3/4}})$. For the "dynamic" prophet inequality, we prove a dynamic policy that returns a prophet inequality of $1 - 1/k$, and this result is tight in the limit. This shows that the upper bound in Hajiaghayi et al. [32] can be improved for two-point distributions once ordering is introduced.

The work in Chapter 3 is done jointly with Shipra Agrawal and Jay Sethuraman.

1.2 Popular Matchings

In the second part of the thesis, we study popular matchings, which originated as a relaxation of stability in stable matchings. Stable matchings were first introduced by Gale and Shapley [26] as a way to match participants with preferences. We are given a bipartite graph with independent sets M and W, which we can think of as men and women. Each man has a ranking of the women, as do the women for the men. We want to find a way to match these men and women that take into consideration their preferences. What we want to avoid in a matching is a man and a woman who prefer each other over their matched partners. Such a matching is *unstable* because the man and woman would break up with their matched partners for each other instead. This man and woman pair is called a *blocking pair* because they prevent the matching from being stable. We can think of stable matchings as matchings that satisfy a certain level of fairness among participants. The Gale-Shapley algorithm has been generalized and applied in many contexts, such as assigning students to schools and residents to hospitals.

One drawback of a stable matching is that the non-existence of blocking pairs limits the cardinality of the matching. In some situations, a primary goal of a central planner may be to match as many participants as possible. For example, in matching students to schools, we may ideally want as many students to attend school as possible. In addition, the existence of a blocking pair may not be as important (for example, if it's difficult for students to transfer schools). We are then motivated by the question: Is there a way to match more participants, while allowing blocking pairs, but still maintaining a semblance of fairness?

Popular matchings are a potential answer to this question. A matching M is popular if for every other matching M', the number of nodes that prefer M to M' are at least as many as the number of nodes that prefer M' to M. In other words, if we ask everyone to vote for either M or M' in a head-to-head contest, M is more *popular* and receives at least as many votes as M'. All stable matchings are shown to be popular and can also be shown to be the smallest cardinality popular matchings. Any larger popular matching is therefore not stable and contains blocking pairs. In addition, the

notion of popularity retains a sense of global fairness - it is, after all, what the majority of the people want. Therefore, the concept of popularity is an answer to our previous question of wanting to match more participants but maintaining a sense of fairness.

We already know how to find *a* popular matching, since all stable matchings are popular, but can we find a popular matching that is not stable? Kavitha [39] gives an algorithm that finds the largest sized popular matchings, which uses the Gale-Shapley algorithm as a subroutine. One can then ask, can we find a popular matching that is not the smallest nor the largest? If we put weights on the edges, can we find a maximum weight popular matching?

In Chapter 4, we study these questions. We construct a framework that is able to prove NP-hardness for a variety of questions concerning popular matchings. We do this by reducing from monotone 3-SAT (3-SAT where each clause has all positive or all negative literals), which we know is NP-hard. For any instance of monotone 3-SAT, we construct a bipartite graph, and show that solving 3-SAT is equivalent to finding a popular matching in the graph we construct. The exact graph we construct is slightly different for each question we're trying to answer. However, using this general framework, we can show that a variety of questions regarding popular matchings is NP-hard. Among them, we show that finding a max-weight popular matching is NP-hard, and is also inapproximable to a factor better than $1/2$. Previous results from [16] provide an algorithm achieving a $1/2$ approximation, hence this factor is actually tight. In addition, perhaps surprisingly, we can show that finding a popular matching that is not smallest or largest is also NP-hard. Among positive results, we provide a tractable algorithm for finding the max weight popular matching on graphs with bounded treewidth.

Chapter 2: Optimal Ordering in the Optimal Stopping Problem

2.1 Introduction

Consider a player who can probe a sequence of n independent random variables $X_1, \ldots, X_n$ with known distributions. After observing (the realized value of) X_i, the player needs to decide whether to stop and earn reward X_i, or reject the reward and probe the next variable X_{i+1}. The goal is to maximize the expected reward at the stopping time. This is an instance of the optimal stopping problem, which is a fundamental problem studied from many different aspects in mathematics, statistics, and computer science, and has found a wide variety of applications in sequential decision making and mechanism design.

When the order in which the random variables $X_1, \ldots, X_n$ are probed is fixed, the optimal stopping strategy can be found by solving a simple dynamic program. Under this strategy, at every step i, the player would compare the realized value of the current random variable X_i to the expected reward (under the optimal strategy for the remaining subproblem) from the remaining variables $X_{i+1}, \ldots, X_n$, and stop if the former is greater than the latter. The celebrated prophet inequalities compare the expected reward of the optimal stopping strategy to $E[\max(X_1, X_2, \ldots, X_n)]$, where the latter can be interpreted as the expected reward of a prophet who can foresee (or 'prophesize') the values of all random variables in advance and therefore always stops at the random variable with maximum value.

A seminal result of Krengel and Sucheston [43] upper bounds the ratio of the prophet's expected reward and that of an optimal stopping strategy by 2, for any arbitrary sequence of n random variables. Furthermore, they show that this bound is tight even for $n = 2$. Samuel-Cahn et al. [51] show that we can achieve an approximation ratio of 2 using simpler stopping strategies (rather than an optimal stopping strategy); subsequent work by Chawla et al. [9] and Kleinberg and Weinberg

[41] identify other stopping strategies that establish the same bound. Prophet inequalities have since been used to design simple sequential posted-price mechanisms that guarantee a constant fraction of the social welfare or revenue achievable by any mechanism (see e.g. [9, 32]).

In this paper, we focus on the relatively less studied question of optimizing the *order* in which the random variables should be probed. Specifically, besides choosing a stopping strategy, if the player is free to choose the order in which the random variables are probed, then which ordering would maximize the expected reward at the stopping time? This question is motivated both by practical considerations and theoretical observations about the optimal stopping problem.

In practice, many decision-making settings allow the player such a choice of ordering. Consider for example, the problem of sequentially interviewing candidates for a position, which is often presented as a canonical example of the optimal stopping problem. Assuming that the decision of hire/no-hire needs to be made for each candidate immediately after the interview, the interviewer wants to stop interviewing on reaching the candidate with the highest quality. In such a setting, the interviewer may have the liberty to decide the ordering in which the candidates are invited for an interview. Intuitively, given statistical information about the quality of each candidate (based on their resume for example), some orderings of the candidates can make the decision problem easier and can ensure higher expected quality of the hired candidate. Here we aim to formalize this intuition by studying the question of optimal ordering, as well as quantifying the potential gains to the expected reward from this additional degree of freedom.

Some insights into the impact of ordering can be obtained by considering prophet inequalities. For an arbitrary ordering, the prophet inequality cannot be improved beyond the approximation factor of 2. On the other hand, a prophet inequality with a much improved factor of ≈ 1.342 has been shown for i.i.d. random variables [1, 12]. This gap between the identical and the non-identical distributions can be narrowed through better ordering. An example is the research of Esfandiari et al. [21] that showed a narrowed prophet inequality with a factor of $e/(e-1) \approx 1.58$ for random ordering (improved recently to ≈ 1.503 by Correa et al. [10])

Optimal orderings have potential to close this gap even further. Consider the special case when

all distributions have supports on two or fewer points (hence-forth referred to as the 2-point case). For arbitrary ordering, the prophet inequality cannot be improved beyond the factor of 2 even in this special case. Here is a simple example (similar to the example in [21]) that demonstrates this limitation. Let X_1 be a random variable that takes values $\{0, 1\}$, with probabilities $\{1 - \epsilon, \epsilon\}$ respectively, and X_2 takes value ϵ with probability 1. Then, the prophet's expected reward is $E[\max(X_1, X_2)] = \epsilon + (1 - \epsilon)\epsilon = 2\epsilon - \epsilon^2$. However, for the ordering (X_2, X_1), the reward earned by the player has an expected value ϵ whether or not X_2 is accepted, yielding only a 2-approximation. On the other hand, for the ordering (X_1, X_2), an expected reward the same as the prophet can be achieved by the following strategy: probe X_1, stop if $X_1 = 1$, otherwise probe X_2. Thus, in this example, the best ordering is better than the worst-case ordering by a factor of 2 and the best ordering attains the same value as the prophet. Furthermore, in this example, choosing a random ordering results in an expected reward of $\approx \frac{3}{2}\epsilon$; and therefore, the best ordering is better than the random ordering by a factor of $4/3$.

These observations motivate our investigation into optimal orderings. The optimal ordering problem has been previously studied in mathematics and statistics literature in the 80's (e.g., [27], [33], and [36]), one notable result is that there is always a static ordering of variables that is optimal [36]. However in this literature has been on analytically characterizing the optimal order for some special structured cases (like Bernoulli and exponential distributions). Our focus is on understanding the computational complexity and devising tractable algorithms. One difficulty in such a study is that the nature of this problem changes significantly depending on the type of distributions considered. For example, when distributions are Bernoulli or exponential, the optimal ordering can be found analytically [33], but, the problem remains nontrivial for uniform distributions, and as we show in this paper, even for distributions with very small support.

Unlike the fixed ordering case, the problem of finding an optimal ordering for optimal stopping cannot be easily solved in polynomial time by dynamic programming. The ordering problem is an instance of the more general *stochastic dynamic programs*. Fu et al. [25] provided a polynomial time approximation scheme (PTAS) for a class of stochastic dynamic programs under the assump-

tion that all the distributions involved are supported on a constant number of points. The ordering problem studied here can be formulated as a problem in their class of stochastic dynamic programs. In fact, the optimal ordering problem is a special case of what they refer to as the committed Pandora's box problem, a variant of the Pandora's box problem [53]. Furthermore, recently Segev and Singla [52] as well as Liu et al. [44] provide an efficient polynomial time approximation scheme (EPTAS) for this problem.

In this work, we delve deeper into the optimal ordering problem when the distributions involved have a small support. What is the computational hardness of this problem? Can the problem be solved to optimality under distributions with very small support (2 or 3)? Does there exist an FPTAS? Given that most interesting counter-examples and lower bounds for prophet inequalities and impact of ordering (some of which were discussed above) have been shown for distributions with just 2-point support, the problem appears to be nontrivial even in these special cases.

Our contributions. We show that the optimal ordering problem is NP-hard even under a special case of 3-point distributions where the highest and lowest points of the support are the same for all the distributions. This is surprising, especially since our optimal ordering problem is a special case of the committed Pandora's box problem, which is a slight variant of the Pandora's box problem. And for the latter, an efficient optimal adaptive strategy is known for arbitrary distributions [53]. In fact, the hardness of the committed Pandora's box problem was not understood before our result, even for arbitrary distributions [25].

Among positive results, we present an FPTAS for a special case of 3-point distributions. We also devise an efficient polynomial time algorithm for finding an optimal ordering in the case of 2-point distributions. Further, we show for the 2-point case, under an optimal ordering the prophet inequality holds with a significantly better approximation factor compared to the worst case ordering.

Our results are summarized as follows:

- **NP hardness for 3-point distributions (Theorem 1 in §2.3).** Through a reduction from the

subset product problem, we show that the optimal ordering problem is NP-hard even when each random variable X_i is restricted to be a 3-point distribution with support on $\{0, m_i, 1\}$ for some $m_i \in (0, 1)$, and $E[X_i|X_i > 0] = E[X_j|X_j > 0]$ for all i, j.

- **Optimal ordering for 2-point distributions (Theorem 2 in §2.4.1).** We present a simple quadratic time algorithm for finding an optimal ordering in the 2-point case.

- **New prophet inequality for 2-point distributions (Theorem 3 in §2.4.2).** We prove that given any set of variables with 2-point distributions, under the *optimal ordering*, the prophet inequality holds with a much improved factor of 1.25 as compared to 2 for arbitrary orderings. And further, our prophet inequality is tight for 2-point distributions. This illustrates the significance of the ability to choose an ordering.

- **FPTAS for 3-point distributions (Theorem 4 in §2.5).** We provide an FPTAS for the optimal ordering problem for the case when each random variable $X_i, i = 1, \ldots, n$ has a three-point distribution with support on $\{a_i, m_i, 1\}$ for some $a_i, m_i \in [0, 1]$.

2.1.1 Related Work

Our work builds on a large body of work, starting from the early work on the classical prophet inequality, on finding an optimal ordering for optimal stopping, and on Pandora's box problem, to the more recent work on the prophet secretary problem and its variations. We briefly survey this literature and position our contributions in the context.

Pandora's box problem and stochastic dynamic programs. The optimal stopping problem considered here is similar in spirit to a well-studied—but substantially easier— problem called "Pandora's box problem" [53]. As in our model, there are n random variables $X_1, X_2, \ldots, X_n$ with distributions known to the decision maker. Also, as in our model, the decision maker is free to probe any random variable at any stage. Unlike in our model, however, in the Pandora's box problem, the decision maker is allowed to choose the value of *any* random variable that has been previously probed; and the feature that makes the problem non-trivial is that a random variable can

be probed only at a (known) cost. [53] presents an "index" policy for the Pandora's box problem and shows that such a policy is optimal; these indices are closely related to Gittins indices for the famous multi-armed bandit problem [28]. Our problem is more directly related to a variant called the *Committed Pandora's box problem* [25]. Similar to our setting, there the decision maker is committed to only choosing the value of the last probed random variable. Committed Pandora's box can be formulated as a problem in the class of stochastic dynamic programs studied in [25], There the authors provide a general PTAS for any such problem when the distributions have support on a constant number of points. Our hardness result for 3-point distributions therefore provides a novel and very strong computational hardness result for such stochastic dynamic programs.

Optimal Ordering for optimal stopping: As mentioned earlier, the problem we consider was studied earlier by [36] and [33]. In [33], the authors provide simple ordering rules for some families of random variables; this includes the case when every random variable X_i is uniformly distributed between 0 and some positive number α_i, and some very specific cases of two-point distributions. Our result on optimal ordering of general two-point distributions requires significantly more work, and will reduce to their results in the specific cases studied there. More importantly, these earlier papers [36, 33] also give examples for which simple rules of thumb—ordering based on mean or variance; stochastic ordering, assuming the variables are all stochastically ordered, etc.—do not work. Our NP-hardness result suggests that such heuristic rules are unlikely to be optimal.

Prophet Inequalities: The work of Krengel and Sucheston [43] generated a lot of interest in developing prophet inequalities and in obtaining simpler rules; an important contribution is the work of Samuel-Cahn et al. [51], who provided a simple rule that achieves the same approximation ratio as the optimal stopping rule. A good summary of this early work on classic prophet inequalities is provided by the survey of Hill [34]. The work of Hajiaghayi et al. [32] and Chawla et al. [9] led to several novel applications of these ideas in designing posted price mechanisms with provably good guarantees on social welfare and revenue. Several richer variants and extensions of the classical prophet inequality setting have since been studied in the literature [41, 24, 49, 50, 2, 20, 19]. A

good overview of this line of work is the survey paper by Lucier [45].

Prophet Secretary Problem: In addition to extending the classical prophet inequality, researchers have also investigated various assumptions of the basic model. As an example, can the approximation ratio be improved if the order in which the random variables are drawn is itself chosen uniformly at random. This model, dubbed the *prophet secretary* problem, has been explored actively since its introduction by Esfandiari et al. [21], who showed that an improved approximation ratio of $e/(e-1) \approx 1.58$ can be achieved. They achieved this by using a sequence of non-increasing thresholds on the random variables: in such a policy, we accept the jth random variable if its value exceeds the threshold T_j, regardless of the identity of the random variable being observed. Subsequent work by Correa et al. [12] showed that the same result can be achieved by threshold policies in which the thresholds depend only on the random variable being observed, but is independent of when it is observed. Somewhat surprisingly, in a recent work, Ehsani et al. [20] show that one can recover this bound using a *single* threshold combined with a carefully chosen (randomized) tie-breaking rule. That the bound of $e/(e-1) \approx 1.58$ is not optimal has been demonstrated in a series of recent papers, first by Azar et al. [4] to approximately 1.576 and then by Correa et al. [10] to approximately 1.503. Interestingly, [10] also prove a lower bound of $(\sqrt{3}+1)/2 \approx 1.366$ on the approximation ratio achievable by any algorithm, even for the case of 2-point distributions.

Order Selection: A few recent papers address the order selection version of the prophet inequality problem as well. The focus is typically not on finding an optimal ordering of the random variables, but on establishing (improved) bounds on the performance of an optimal ordering relative to that of a prophet. For example, Yan [54] proves a bound of $e/(e-1)$ for this performance and later Beyhaghi et al. [5] improves the bound to 1.528. Note that any bound on the prophet secretary problem (i.e. random order) automatically carries over to this case. Interestingly, our approximation ratio of 1.25 for the case of 2-point distributions *cannot* be achieved for the prophet secretary version of the problem. This is implied by our simple example presented in the introduction, as well as by the lower bound by Correa et al. [10].

IID Instances: We close by mentioning the recent developments on prophet inequalities for the

case in which all the random variables are drawn from the same distribution. Note that order selection is irrelevant in this case, so that all three versions of the classic problem—worst case order, random order, and best case order—coincide. Hill [35] constructed a worst-case family of instances for each n and later showed that the approximation ratio for these instances is at least ≈ 1.342; this has been shown to be tight in a recent paper of Correa et al. [12]. Interestingly, we do not know if the worst-case instances for the order selection problem are i.i.d. instances.

2.2 Preliminaries

Let $X_1, \ldots, X_n$ be random variables with (known) distributions $D_1, \ldots, D_n$ respectively. An ordering is defined by a permutation σ of indices $\{1, \ldots, n\}$.

The optimal stopping problem under a fixed ordering σ is defined as follows. At each time step $t = 1, \ldots, n$, a player first observes the value of $X_{\sigma(t)}$, generated independently from distribution $D_{\sigma(t)}$. Then, the player can either decide to stop and accept reward $X_{\sigma(t)}$; or reject $X_{\sigma(t)}$ and continue to round $t + 1$. The stopping time τ is defined as the time step t at which the player stops and accepts reward $X_{\sigma(t)}$. If the player reaches the end of the sequence without accepting any reward, then the game stops automatically, and the stopping time is defined as $\tau = n + 1$ with reward $X_\tau := 0$ (as a consequence, the player should never accept a negative realization from a random variable). The goal is to maximize the expected reward $E[X_\tau]$ at the stopping time τ, where the expectation is taken both over the stopping time (which may depend on the random instantiations of the past $X_{\sigma(t)}, t = 1, \ldots, \tau - 1$) and the distribution of X_τ. We denote the expected reward at the optimal stopping time as V_σ. That is,

$$V_\sigma := E[X_{\tau^*}] \tag{2.1}$$

where τ^* is the stopping time given by the optimal stopping rule that maximizes $E[X_\tau]$.

It is easy to see that, due to the optimal substructure property in this problem, the optimal stopping rule is defined by a simple Dynamic Program (DP). Specifically, let $V_\sigma(j)$ denote the

optimal expected reward for the subsequence $X_{\sigma(j)}, \ldots, X_{\sigma(n)}$, so that $V_\sigma = V_\sigma(1)$. Then, for $j = 1, \ldots, n$

$$V_\sigma(j) = E[\max(X_{\sigma(j)}, V_\sigma(j+1))], \text{ with} \tag{2.2}$$

$$V_\sigma(n+1) = 0. \tag{2.3}$$

The following additional notation will be useful later in the proofs: for any sequence of random variables, say $S := (X_1, \ldots, X_n)$, we define $V(X_1, \ldots, X_n)$ (also denoted as $V(S)$) as

$$V(X_1, \ldots, X_n) := E[\max(X_1, E[\max(X_2, \cdots, E[\max(X_{n-1}, E[\max(X_n, 0)]) \cdots])])] \tag{2.4}$$

Furthermore, if
$S = (X_1, \ldots, X_k)$ and $T = (X_{k+1}, \ldots, X_m)$ are two disjoint sequences of random variables, then we denote $V(S, T) := V(X_1, \ldots, X_m)$. Then, note that for any ordering σ and $1 \leq j < i \leq w$,

$$V_\sigma(j) = V(X_{\sigma(j)}, \ldots, X_{\sigma(n)}) = V(X_{\sigma(j)}, \ldots, X_{\sigma(i)}, V_\sigma(i+1)) \tag{2.5}$$

Given the above DP equations, the values $V_\sigma(1), \ldots, V_\sigma(n)$ can be calculated by backward induction. Then, the optimal stopping policy is defined as follows: at any time step t, compare the realized value of the random variable $X_{\sigma(t)}$ to $V_\sigma(t+1)$; if this realized value is at least $V_\sigma(t+1)$, stop and accept the reward; otherwise, if $t < n$, continue to probe $X_{\sigma(t+1)}$. Below we are ready to define the optimal ordering problem.

Definition 2.2.1 (Optimal ordering for optimal stopping problem)**.** *For any given ordering σ of n random variables $X_1, \ldots, X_n$, let V_σ be the expected value at the optimal stopping time as defined in* (2.1)*. We define the problem of optimal ordering for optimal stopping as the problem of choosing an ordering σ that maximizes V_σ i.e., the problem of finding*

$$\sigma^* = \arg\max_\sigma V_\sigma. \tag{2.6}$$

Note that our definition of the optimal ordering problem is restricted to finding an optimal *static* ordering. In other words, the ordering of all the random variables is decided in advance based only on the distributions, and the observed values of the random variables examined up to a particular stage are not used to dynamically change the ordering of the remaining variables. This is without loss of generality due to a result from *Hill [36, Theorem 3.11] that there is always a static ordering of the variables that is optimal*.

k-point distributions. In this paper, we study computational hardness and present algorithmic methods for the optimal ordering problem. We investigate the complexity of this problem for the case when the distributions have a finite k-point support. Specifically, we consider two-point and three-point distributions.

Definition 2.2.2 (Two-point distributions)**.** *A random variable X_i with a two-point distribution is defined by three parameters a_i, b_i, p_i, and takes value*

$$X_i = \begin{cases} a_i, & w.p.\ 1 - p_i, \\ b_i, & w.p.\ p_i. \end{cases}$$

Here, $a_i \leq b_i$ will be referred to as the left and the right end-point of the support, respectively.

Definition 2.2.3 (Three-point distributions)**.** *A random variable X_i with a three-point distribution is defined by five parameters a_i, m_i, b_i, p_i, q_i, and takes value*

$$X_i = \begin{cases} a_i, & w.p.\ 1 - p_i - q_i, \\ m_i, & w.p.\ p_i, \\ b_i, & w.p.\ q_i. \end{cases}$$

Here, $a_i \leq m_i \leq b_i$, will be referred to as the left end-point, the middle point, and the right end-point of the support, respectively.

2.3 Hardness of optimal ordering

We show that the problem of finding an optimal ordering of the random variables is NP-hard even in the highly restricted special case where each random variable X_i is supported on exactly three points: $a_i = 0$, $b_i = 1$, and $m_i \in (0, 1)$. Let $q_i := P(X_i = 1)$ and $p_i := P(X_i = m_i)$, so that $P(X_i = 0) = 1 - p_i - q_i$. We also assume that $p_i > 0$, $q_i > 0$, and $p_i + q_i < 1$ for all i, so that each random variable can assume each value in its support with positive probability.

Theorem 1. *The problem of optimal ordering for optimal stopping (refer to Definition 2.2.1) is NP-hard for the case when for each $i = 1, \ldots, n$, the random variable X_i has a three-point distribution with support on $\{0, m_i, 1\}$ for some $m_i \in (0, 1)$.*

To prove the hardness result, we first prove some useful results on the structure of optimal orderings and the optimal stopping rules for such random variables.

Fix any ordering σ of the random variables $X_1, X_2, \ldots, X_n$. In this case, the optimal stopping policy essentially partitions the n variables into two categories: those which will be accepted on being probed if and only if they realize their right endpoint 1, i.e., the (ordered) subset $S^\sigma := \{X_{\sigma(i)}, i \in [n] : V_{\sigma(i+1)} > m_{\sigma(i)}\}$; and the remaining (ordered) subset $T^\sigma := \{X_{\sigma(1)}, \ldots, X_{\sigma(n)}\} \setminus S^\sigma$. Note that since the last random variable will always be accepted (if probed) irrespective of its value, the last variable is always in T^σ.

We claim that in *any* optimal ordering σ^*, the random variables in S^{σ^*} must appear before the random variables in T^{σ^*}. Further, the random variables in T^{σ^*} must be ordered in a weakly descending order of E_i, where

$$E_i := E[X_i | X_i > 0] \;=\; \frac{m_i p_i + q_i}{p_i + q_i}$$

Claim 2.3.1. *Given random variables $X_1, \ldots, X_n$ that have three-point distributions with support $\{0, m_i, 1\}$,, and probabilities $\{1 - p_i - q_i, p_i, q_i\}$ such that $m_i \in (0, 1), p_i > 0, q_i > 0, p_i + q_i < 1$, for $i = 1, \ldots, n$. Then, an ordering σ is optimal only if (i) S^σ precedes T^σ; (ii) the random*

variables in S^σ are arranged arbitrarily; and (iii) the random variables in T^σ appear in a weakly descending order of E_i. In particular, if $E_1 = E_2 = \ldots = E_n$, then the random variables in T^{σ^*} can be arranged arbitrarily as well. Further, for any ordering σ satisfying (i)-(iii), if the unordered sets S^σ, T^σ are same as the unordered sets $S^{\sigma^*}, T^{\sigma^*}$ for some optimal ordering σ^*, then σ is also optimal.

Proof. □

Proof. A complete proof of this lemma is provided in Section A.1. The proof uses an interchange argument, where we show that if in an optimal ordering σ^*, the ordering of any pair of variables X_i, X_j violates the stated conditions, then they can be interchanged to increase V_{σ^*} which would be a contradiction to the optimality of the ordering σ^*. □

Remark 2.3.2. *In fact, if the conditions $m_i \in (0,1), p_i > 0, q_i > 0, p_i + q_i < 1$ are not satisfied, e.g., if there are some variables with $m_i \in \{0,1\}$, or if the probability of one or more of the three support points is 0, then the proof of Claim 2.3.1 can be modified to show that the conditions (i), (ii), (iii) are still sufficient (though not necessary) for an ordering σ to be optimal.*

Definition 2.3.3 (Ordered partitions & Optimal partitioning problem)**.** *A sequence (S,T) of n random variables is an ordered partition if S, T partition the set of variables $X_1, \ldots, X_n$, T is non-empty, the variables within S are ordered arbitrarily, and the variables within T are ordered in a weakly descending order of E_i. The optimal partitioning problem is defined as the problem of finding an ordered partition (S,T) that maximizes $V(S,T)$ among all ordered partitions.*

Corollary 2.3.4. *A corollary of Claim 2.3.1 is that in the three-point distribution case considered here, the optimal ordering problem is equivalent to the optimal partitioning problem.*

We prove the NP-hardness of the optimal ordering problem by showing that finding an optimal partition is NP-hard. To that end, we consider the problem SUBSET PRODUCT, which is a multiplicative analog of the SUBSET SUM problem, and is known to be NP-complete (see [47]):

Problem 1. SUBSET PRODUCT*: Given integers $a_1, \ldots, a_n$ with each $a_i > 1$ and a positive integer B, is there a subset $T \subseteq N$ such that $\prod_{i \in T} a_i = B$?*

Proposition 2.3.5. *The optimal partitioning problem (refer to Definition 2.3.3) is NP-hard when each of random variables $X_1, \ldots, X_n$ have three-point distributions with support $\{0, m_i, 1\}$, and probabilities $\{1 - p_i - q_i, p_i, q_i\}$ such that $m_i \in (0, 1), p_i > 0, q_i > 0, p_i + q_i < 1$, for $i = 1, \ldots, n$.*

Proof. Proof: Given an instance of SUBSET PRODUCT, consider the following collection of random variables. Associated with each element a_i in the subset product problem is a random variable X_i with distribution shown in Table 2.1.

Value	0	$\frac{B^2-a_i}{B^2+1}$	1
Probability	$\frac{1}{a_i^2}$	$\frac{a_i-1}{a_i^2}$	$\frac{a_i-1}{a_i}$

Table 2.1: Distribution of X_i

Notice that the X_i has support 0, and 1, and $m_i = (B^2 - a_i)/(B^2 + 1)$ and probabilities $p_i = \frac{a_i-1}{a_i^2}, q_i = \frac{a_i-1}{a_i}$. Notice also that because $a_i > 1$ for all i, we have that $m_i \in (0, 1)$, $0 < p_i < 1$, $0 < q_i < 1$ and $p_i + q_i < 1$ for all i. Finally, observe that

$$E_i \;:=\; E[X_i | X_i > 0] \;=\; \frac{(\frac{B^2-a_i}{B^2+1})(\frac{a_i-1}{a_i^2}) + \frac{a_i-1}{a_i}}{1 - \frac{1}{a_i^2}} \;=\; \frac{B^2}{B^2+1},$$

which is independent of i. Thus, in any ordered partition (S, T) for this instance, the ordering within S and the ordering within T is irrelevant.

To avoid cumbersome notation, we let S and T denote a partition of the indices $\{1, 2, \ldots, n\}$. The expected reward $V(S, T)$ for an ordered partition (S, T) can be written as

$$V(S,T) \;=\; 1 - \Pi_{i \in S}(1 - q_i) + \Big(\Pi_{i \in S}(1 - q_i)\Big)\Big(1 - \Pi_{j \in T}(1 - p_j - q_j)\Big)\frac{B^2}{B^2+1} \qquad (2.7)$$

An easy way to see why is to exploit the irrelevance of the relative ordering within S and T: the decision maker earns 1 whenever any random variable in S is observed to take on a value of 1; if none of the random variables in S is accepted, the conditional expected value of any accepted random variable in T is the same, and this possibility occurs unless every one of the random variables in T is observed to be zero. Using the fact that $q_i = 1 - 1/a_i$ and $p_i = 1/a_i^2 - 1/a_i$, we can rewrite Eq. (2.7) as

$$V(S,T) \;=\; 1 - \prod_{i\in S}\frac{1}{a_i} + \left(\prod_{i\in S}\frac{1}{a_i}\right)\left(1 - \prod_{i\in T}\frac{1}{a_i^2}\right)\left(\frac{B^2}{B^2+1}\right).$$

Let $\gamma := \prod_{i=1}^{n} a_i$, $\gamma_T := \prod_{i\in T} a_i$, $\gamma_S := \prod_{i\in S} a_i$. Then $\gamma_S = \gamma/\gamma_T$ and $V(S,T)$ can be written solely as a (one-dimensional) function of γ_T as follows:

$$V(S,T) = f(\gamma_T) := 1 - \frac{\gamma_T}{\gamma} + \frac{\gamma_T}{\gamma}\left(1 - \frac{1}{\gamma_T^2}\right)\frac{B^2}{B^2+1}$$

Differentiating $f(\cdot)$ with respect to γ_T twice, we see that

$$f'(\gamma_T) = -\frac{1}{\gamma} + \left(\frac{B^2}{B^2+1}\right)\left(\frac{1}{\gamma} + \frac{1}{\gamma\gamma_T^2}\right)$$

and

$$f''(\gamma_T) = \frac{-2B^2}{\gamma\gamma_T^3(B^2+1)} < 0$$

Thus, $f(\gamma_T)$ is strictly concave in γ_T and achieves its maximum when $f'(\cdot) = 0$, which occurs when $\gamma_T = B$.

To complete the argument, we observe the following: given any instance of SUBSET PRODUCT, we construct the corresponding instance of our optimal partitioning problem and solve it to optimality. The optimal partition (S,T) has $\gamma_T = B$ if and only if the given instance of SUBSET PRODUCT is a "yes" instance. Thus, the NP-completeness of SUBSET PRODUCT implies the NP-hardness of our optimal partitioning problem. □

2.4 Ordering two-point distributions

In this section, we investigate optimal ordering and its significance for the case when all random variables $X_i, i = 1, \ldots, n$ have two point distributions as defined in Definition 2.2.2. Recall that under two point distribution, a random variable X_i can take two possible values $\{a_i, b_i\}$ (w.l.o.g. $a_i \leq b_i$), with probability of the left end-point a_i and the right end-point b_i being $1 - p_i$ and p_i, respectively.

Our first result is the design of a simple and efficient algorithm for finding an optimal ordering in this case. Next, to illustrate the significance of being able to choose an ordering in this case, we prove a prophet inequality with an improved factor of 1.25 for optimal ordering, as compared to the factor of 2 for the worst-case ordering. We also show that this bound is tight, i.e., there exist two-point distributions under which even an optimal ordering cannot achieve an approximation ratio better than 1.25 relative to the reward of the prophet.

We mention that the optimal order for several special cases of two-point distributions were proven in Theorem 4.6 in [33]. Specifically, they prove that the optimal order for the following families of random variables is by decreasing α_i, where α_i is defined in one of the following ways: (1) X_{α_i} is a Bernoulli random variable with parameter $\alpha_i \in [0, 1]$; (2) X_{α_i} is equally likely to be α_i or $-\alpha_i$; (3) X_{α_i} is α_i with probability $1/\alpha_i$ and 0 otherwise; or (4) X_{α_i} is equally likely to be α_i or $\alpha_i + 1$.

Here, we provide a much more general result by presenting an $O(n^2)$ algorithm for finding the optimal ordering given any collection of arbitrary two-point distributions. We also provide structural insights into the optimal ordering for arbitrary 2-point distributions.

2.4.1 Optimal ordering algorithm

Theorem 2. *Given random variables $X_i, i = 1, \ldots, n$ with arbitrary two-point distributions, there exists an algorithm to find the optimal ordering for optimal stopping in $O(n^2)$ time.*

To derive the above result, we first investigate a simpler case when the left endpoint $a_i = 0$ for

all i. In this case, the optimal ordering turns out to be very simple: the variables can be ordered simply in the descending order of their right endpoints. Our key result is in Lemma 2.4.2 and Corollary 2.4.3 where we show a Left Support Property (LSP) of optimal ordering for any set of bounded support distributions. This property is used to extend the above simple algorithm and obtain an optimal ordering algorithm for general two-point distributions.

The following lemma characterizes the optimal ordering for the case when $a_i = 0$ for all i.

Lemma 2.4.1. *Given random variables $X_i, i = 1, \ldots, n$ that are two-point distributions with $a_i = 0, i = 1, \ldots, n$. Then, an optimal ordering can be obtained by ordering the variables in a descending order of their right endpoints b_i. That is, an ordering σ is optimal if $b_{\sigma(1)} \geq \cdots \geq b_{\sigma(n)}$. Furthermore, under an additional condition that $a_i \neq b_i, 0 < p_i < 1$ for all i, an ordering σ is optimal only if $b_{\sigma(1)} \geq \cdots \geq b_{\sigma(n)}$.*

Proof. Proof: Given two-point distributions with $a_i = 0$ and an ordering σ with $b_{\sigma(1)} \geq \cdots \geq b_{\sigma(n)}$, the optimal stopping policy (refer to Section 2.2) reduces to the following: at any step t, check if $X_{\sigma(t)} = b_{\sigma(t)}$ (i.e., realizes its right endpoint); if yes, stop; otherwise continue. Since the variables are probed in descending order of right endpoints, the expected value V_σ for this policy is simply equal to the maximum of the right endpoints realized, or 0. Since each X_i can take either value $a_i = 0$ or $b_i \geq 0$, this is same as $E[\max(X_1, \ldots, X_n)]$, that is, the expected hindsight maximum or the prophet's expected reward. Thus, trivially such an ordering σ is optimal.

To prove the 'only if' part of the lemma statement, suppose that for an ordering σ, there exists j such that $b_{\sigma(j)} < b_{\sigma(j+1)}$. From the above discussion, there exists an ordering that achieves the expected hindsight maximum reward. Hence, it is sufficient to prove that with some positive probability, the reward at the stopping time under ordering σ will be strictly smaller than the hindsight maximum.

Consider the event that $X_{\sigma(j)}$ is probed and takes value $b_{\sigma(j)}$. This event will happen, e.g., if $X_{\sigma(i)} = 0$ for all $i < j$ and $X_{\sigma(j)} = b_{\sigma(j)}$, which has a positive probability since $0 < p_i < 1$ for all i. Under this event, there are two possible scenarios for any stopping policy: either the stopping policy can accept $X_{\sigma(j)} = b_{\sigma(j)}$ and stop; or it can continue to probe the next variable.

In both scenarios, we argue there is a positive probability that the hindsight reward is higher than the reward at the stopping time. In the first scenario, this is the case if $X_{\sigma(j+1)} = b_{\sigma(j+1)} > b_{\sigma(j)}$ which can happen with probability $p_{\sigma(j+1)} > 0$. In the second scenario, this is the case if $X_{\sigma(j+1)} = \cdots = X_{\sigma(n)} = 0$, which can happen with probability $\prod_{i \geq j+1}(1 - p_{\sigma(i)}) > 0$.

□

Lemma 2.4.2 (Left Support Property (LSP))**.** *Suppose X_1, X_2, …, X_n are random variables with bounded support $X_i \in [a_i, b_i]$. Then there exists an optimal ordering σ with the property that $\Pr(X_{\sigma(i)} \leq V_\sigma(i+1)) > 0$ for $i = 1, \ldots, n-1$. That is, for every X_i, its distribution has non-zero support on the left of the value $V_\sigma(i+1)$. We refer to this property as the Left Support Property (LSP)*

Proof. Proof: The proof is by construction. We show that from any optimal ordering we can obtain an ordering that satisfies LSP, without decreasing its value. W.l.o.g. let $\sigma = (1, \ldots, n)$ be an optimal ordering that does not satisfy LSP. Let X_i be the first r.v. among $X_1, \ldots, X_{n-1}$ which violates LSP, i.e. $P(X_i \leq V_\sigma(i+1)) = 0$. Then, $X_i > V_\sigma(i+1)$ with probability 1, and by definition

$$V_\sigma(i) = E[\max(X_i, V_\sigma(i+1))] = E[X_i].$$

Now, consider an alternate ordering

$$\sigma' = (1, \ldots, i-1, i+1, \ldots, n, i),$$

where variable X_i is pushed to the end. We show that σ' has the same value as σ and satisfies LSP.

Observe that (refer to (2.5)),

$$V_{\sigma'}(i) = V(X_{i+1}, \ldots, X_n, X_i) = E[\max(X_{i+1}, \cdots, E[\max(X_n, E[\max(X_i, 0)])] \cdots)] \geq E[X_i]$$

Thus,

$$V_{\sigma'}(i) \geq V_\sigma(i) = E[X_i] \tag{2.8}$$

From above we can derive the following conclusions about the new ordering σ':

- LSP property is satisfied for indices $i, \ldots, n-1$ in σ': Suppose for contradiction that for some $i \leq j \leq n-1$, LSP property is violated, i.e., suppose that $X_{\sigma'(j)} > V_{\sigma'}(j+1)$ with probability 1. Now, since

$$V_{\sigma'}(j+1) \geq V_{\sigma'}(n) = E[\max(X_i, 0)] \geq E[X_i],$$

  we have that $X_{\sigma'(j)} > E[X_i]$ with probability 1. Since $\sigma'(j) \in \{i+1, \ldots, n\}$, this implies $V_\sigma(i+1) = V(X_{i+1}, \ldots, X_n) > E[X_i]$. For its expected value to be below $V_\sigma(i+1)$, X_i must take value below $V_\sigma(i+1)$ with non-zero probability. This implies that LSP property is satisfied by X_i in ordering σ, which is a contradiction to the assumption we started with.

- LSP property is satisfied for indices $1, \ldots, i-1$ in σ': Since $\sigma(j) = \sigma'(j)$ for $1 \leq j \leq i-1$, and $V_{\sigma'}(i) \geq V_\sigma(i)$, we have that (refer to (2.5))

$$V_{\sigma'}(j) = V(X_j, \ldots, X_{i-1}, V_{\sigma'}(i)) \geq V(X_j, \ldots, X_{i-1}, V_\sigma(i)) = V_\sigma(j).$$

  This means for $1 \leq j \leq i-1$,

$$\Pr(X_{\sigma'(j)} \leq V_{\sigma'}(j+1)) \geq \Pr(X_{\sigma(j)} \leq V_\sigma(j+1)) > 0,$$

  where the last inequality followed from the assumption that X_i is the first variable to violate LSP in the ordering σ. Thus, the LSP property is satisfied by $1 \leq j \leq i-1$ in the new ordering.

- $V_{\sigma'} \geq V_\sigma$: If $i = 1$, $V_{\sigma'}(1) \geq V_\sigma(1)$ follows from (2.8), otherwise it follows from the observation in the previous bullet that $V_{\sigma'}(j) \geq V_\sigma(j)$ for $1 \leq j \leq i-1$.

Thus, the new ordering σ' satisfies LSP property, and has value greater than or equal to optimal ordering ($V_{\sigma'} \geq V_\sigma$). In fact, since σ is optimal, $V_{\sigma'} = V_\sigma$. □

A corollary of Lemma 2.4.2 is the following Left Endpoint Property (LEP) for two-point distributions:

Corollary 2.4.3 (Left Endpoint Property (LEP) for two-point distributions)**.** *Suppose X_1, X_2, ..., X_n are random variables with two-point support $X_i \in \{a_i, b_i\}$. Then there exists an optimal ordering σ with the property that $a_{\sigma(i)} \leq V_\sigma(i+1)$ for $i = 1, \ldots, n-1$.*

The left endpoint property allows us to derive the following characterization of the optimal ordering in two-point distributions, which will significantly reduce the space of orderings to search over in order to find the optimal ordering.

Proposition 2.4.4. *Given any collection of n random variables with two-point distributions, define n orderings as follows: for each $i = 1, \ldots, n$, define σ^i as an ordering obtained by setting the last variable as X_i, and order the remaining variables in a weakly descending order of their right endpoints. Then, at least one of these n orderings is optimal.*

Proof. Proof: By Corollary 2.4.3 there exists an optimal ordering satisfying LEP, w.l.o.g. assume it is $\sigma^* = (1, \ldots, n)$. Let σ be an ordering such that $\sigma(n) = n$, and $b_{\sigma(1)} \geq \ldots \geq b_{\sigma(n-1)}$, i.e., σ is one of the n orderings defined in the proposition statement. Then, we show that $V_\sigma \geq V(X_1, \ldots, X_n)$.

W.l.o.g., we assume that $b_i > E[\max\{X_n, 0\}]$ for all i, otherwise, (any X_i not satisfying this will always be rejected in both σ and σ^* and can be ignored completely in this argument). We will show that $V_\sigma = V(X_{\sigma(1)}, \ldots, X_{\sigma(n-1)}, X_n) \geq V(X_1, \ldots, X_{n-1}, X_n) = V_{\sigma^*}$.

For $i = 1, \ldots, n-1$, define:

$$X_i' = \left\{ \begin{array}{ll} E[\max(X_n, 0)], & \text{w.p. } 1 - p_i, \\ b_i, & \text{w.p. } p_i \end{array} \right\}$$

and

$$X_i'' = \left\{ \begin{array}{ll} 0, & \text{w.p. } 1 - p_i, \\ b_i - E[\max(X_n, 0)], & \text{w.p. } p_i \end{array} \right\}$$

We claim the following sequence of relations:

$$
\begin{aligned}
V(X_{\sigma(1)}, \ldots, X_{\sigma(n-1)}, X_n) &\geq V(X'_{\sigma(1)}, \ldots, X'_{\sigma(n-1)}, X_n) && (1)\\
&= V(X'_{\sigma(1)}, \ldots, X'_{\sigma(n-1)}) && (2)\\
&= V(X''_{\sigma(1)}, \ldots, X''_{\sigma(n-1)}) + E[\max(X_n, 0)] && (3)\\
&\geq V(X''_1, \ldots, X''_{n-1}) + E[\max(X_n, 0)] && (4)\\
&= V(X'_1, \ldots, X'_{n-1}) && (5)\\
&= V(X'_1, \ldots, X'_{n-1}, X_n) && (6)\\
&= V(X_1, \ldots, X_n) && (7)
\end{aligned}
$$

We now justify each of the relations:

For (1): First notice that $V_\sigma(i+1) \geq E[\max(X_n, 0)]$ for all $i < n$. If $a_{\sigma(i)} \leq E[\max(X_n, 0)] \leq V_\sigma(i+1)$, then $X_{\sigma(i)}$ would be rejected if its left endpoint is realized, therefore, increasing its left endpoint to $E[\max(X_n, 0)]$ does not change the overall expected value. On the other hand, if $a_{\sigma(i)} > E[\max(X_n, 0)]$, then transforming by decreasing the left endpoint from $a_{\sigma(i)}$ to $E[\max\{X_n, 0\}]$ can only decrease (or not change) the overall expected reward.

For (2) and (6): If for any of the two orderings, the second last variable ($X'_{\sigma(n-1)}$ or X'_{n-1}) is probed and its left endpoint ($E[\max(X_n, 0)]$) is realized, then accepting it would give $E[\max(X_n, 0)]$ reward, while rejecting it and continuing would also give $E[\max(X_n, 0)]$. Thus, removing X_n does not affect the overall expected reward.

For the (3) and (5): $X'_i \geq 0$ and $X''_i = X'_i - E[\max(X_n, 0)] \geq 0$ due to the assumption made (w.l.o.g.) that $b_i \geq E[\max(X_n, 0)]$; we can therefore use an observation that for any sequence of variables $(Y_1, \ldots Y_k)$ and constant c if $Y_i + c \geq 0$, then $V(Y_1 + c, \ldots Y_k + c) = V(Y_1, \ldots Y_k) + c$. This is formally proven in Lemma A.4.1 in the appendix.

For (4): Since X''_i are two-point distributions with 0 left endpoint, it follows from Lemma 2.4.1 since σ is an optimal ordering.

For (7): This is due to the fact that σ^* is an LEP ordering. Therefore, $a_i \leq V_{\sigma^*}(i+1)$ for all $i < n$, so that if X_i realizes its left endpoint, it will be rejected anyway. Hence, changing its left endpoint to $E[\max(0, X_n)] \leq V_{\sigma^*}(i+1)$ will not change the overall expected value.

□

Proof. Proof of Theorem 2 Proposition 2.4.4 narrows down the space of orderings to be searched over to just the n orderings $\sigma^i, i = 1, \ldots, n$, where the ordering σ^i is as defined in Proposition 2.4.4. It will take $O(n)$ time to compute the expected reward V_{σ^i} of each of these n orderings, and therefore takes $O(n^2)$ time to compute the expected reward for all n orderings and find the best ordering. □

2.4.2 Prophet inequality for optimal ordering

In the previous subsection, we presented an an algorithm for finding the optimal ordering for two-point distributions. In this section, we prove a prophet inequality upper bounding the ratio of the prophet's reward and the expected reward under best ordering by 1.25. In comparison, as noted in 2.1.1, there exist (two-point) distributions for which this factor is 2 under worst-case ordering [51]. This illustrates the potential benefits of ordering.

Theorem 3 (Prophet inequality for two-point distributions)**.** *Given any set of n random variables $X_1, \ldots, X_n$ with two-point distributions, the prophet's expected reward is within* 1.25 *factor of the expected reward at stopping time under optimal ordering, i.e.,*

$$\frac{E[\max(X_1, \ldots, X_n)]}{(\max_\sigma V_\sigma)} \leq 1.25 \tag{2.9}$$

where V_σ is as defined in (2.1).

Proof. Proof: Let $X_1, \ldots, X_n$ be a set of random variables with two-point distributions, where X_i takes values $\{a_i, b_i\}$, $a_i \leq b_i$, with probabilities $1 - p_i$ and p_i, respectively. Let $X^* := X_{i^*}$ be the random variable with largest left endpoint, i.e., $i^* = \arg\max_i a_i$, with support points and probability

denoted as $\{a^*, b^*, p^*\} = \{a_{i^*}, b_{i^*}, p_{i^*}\}$. Let $U := \{i : b_i \geq b^*\} \backslash i^*$, $W := \{i : b^* > b_i \geq a^*\}$. Let the variables in U, W be ordered in weakly descending order of their right endpoints b_i.

Note that by definition of U, W for any $i \notin U \cup W \cup \{i^*\}$, the right endpoint b_i must be strictly smaller than the left endpoint a^* of X^*. Thus, such an X_i will always take a value smaller than $\max(X_1, X_2, \ldots, X_n)$ so that $E[\max(X_1, \ldots, X_n)] = E[\max_{i \in U \cup W \cup \{i^*\}} X_i]$. Further, for any ordering σ, let $\bar{\sigma}$ be the ordering restricted to the subset of variables in $U \cup W \cup \{i^*\}$; then $V_\sigma \geq V(X_{\bar{\sigma}})$, where $X_{\bar{\sigma}}$ denotes the variables in $U \cup W \cup \{i^*\}$ ordered according to $\bar{\sigma}$. Thus, ignoring the remaining variables $\{i \notin U \cup W \cup \{i^*\}\}$ can only hurt the prophet inequality, since the numerator would remain the same while the denominator can only decrease. We therefore ignore such variables in the remaining discussion, and assume that $U \cup W \cup \{i^*\} = \{1, \ldots, n\}$.

Next, we prove the prophet inequality (2.9) by showing that the expected reward at the stopping time under one of the following two orderings is at least $4/5$ of the expected hindsight maximum.

1. Ordering $\sigma^1 = (U, W, i^*)$. In this ordering, we first have the variables with right endpoint larger than b^*, in (weakly) decreasing right endpoint order. Then, the variables with right endpoint smaller than b^* (but larger than a^*), in (weakly) decreasing right endpoint order. And, finally we have i^*.

2. Ordering $\sigma^2 = (U, i^*, W)$. That is, essentially the variables are ordered in (weakly) decreasing right endpoint order.

An easy scenario is when at least one of the variables in $i \in U$ realizes its right endpoint b_i. In this scenario, under both the above orderings $\sigma \in \{\sigma^1, \sigma^2\}$, the optimal stopping policy will achieve the same reward as the prophet. To see this, first note that since

$$\min_{i \in U} b_i \geq b^* \geq \max_{i \in \{V \cup i^*\}} b_i \geq V_\sigma(|U| + 1),$$

the variables in $W \cup \{i^*\}$ will never be probed in this scenario. Among those in U, the optimal stopping policy will reject all variables in U whose left endpoint is realized, and accept the first

variable whose right endpoint is realized. This is because, $a^* \geq a_i$ for all $i \in U$, so that $V_\sigma(|U|+1) \geq a^* \geq a_i$, and because the variables in U are ordered in decreasing right endpoint order. Therefore, the optimal stopping policy will achieve a reward of $\max_{i \in U: X_i = b_i} b_i$. This will also be the prophet's reward, as the maximum value can never be (uniquely) achieved by some left endpoint of U (because $a_i \leq a^*, \forall i \in U$) and none of the variables in $W \cup \{i^*\}$ can achieve a higher value than $\max_{i \in U: X_i = b_i} b_i$.

In the remaining proof we consider the alternate scenario that every variable in $i \in U$ realizes its left endpoint a_i. Furthermore, we assume that W is non-empty, otherwise the reward for both the prophet and the optimal stopping policy will simply be X^*. Since $a_i \leq a^*$ for all i, in this scenario, the prophet's reward is $\max_{i \in W \cup \{i^*\}} X_i$; and since $V_\sigma(|U|+1) \geq a^*$, the optimal stopping policy for both orderings $\sigma \in \{\sigma^1, \sigma^2\}$ will reject all the variables in U. Thus, the variables in U do not effect the reward of either the prophet or at the stopping time, and we ignore them for the remaining discussion and focus on the reward obtained from the variables in $W \cup \{i^*\}$.

Under the two orderings σ^1, σ^2, the random variables are ordered as (W, i^*) and (i^*, W), respectively, so that $V_{\sigma^1} = V(X_W, X^*)$ and $V_{\sigma^2} = V(X^*, X_W)$, where X_W denotes the sequence of random variables in W.

Now, define quantities

$$b_w := E[\max_{i \in W} X_i | \exists i \in W : X_i = b_i], \text{ and } p_w := \Pr(\exists i \in W : X_i = b_i) = 1 - \prod_{i \in W}(1 - p_i).$$

Then, we can make the following observations about the prophet's expected reward and the maximum expected reward under the orderings σ^1, σ^2.

Firstly, since $b^* \geq b_w \geq a^*$, the prophet's maximum reward is given by

$$MAX := E[\max(X_W, X^*)] = p^* b^* + (1 - p^*) p_w b_w + (1 - p^*)(1 - p_w) a^* \tag{2.10}$$

For the ordering σ^1, since $V(X^*) = \mu^* := p^* b^* + (1 - p^*) a^*$ and variables in W are ordered in decreasing right endpoint, the optimal stopping policy will either accept some X_i in W with the

largest realized right endpoint b_i as long as it is more than μ^*, or wait till i^* to get expected reward μ^*, so that the expected reward

$$V_{\sigma^1} = V(X_W, \mu^*) \geq E[\max(\max_{i \in W: X_i = b_i} b_i, \mu^*)] \geq p_w b_w + (1 - p_w)\mu^*$$

For the ordering σ^2, since the variables in W are ordered in decreasing right endpoint, when probing X_W, the optimal stopping policy will accept the first variable that realizes its right endpoint, so that $V(X_W) \geq p_w b_w$ and

$$V_{\sigma^2} = V(X^*, X_W) \geq E[\max(X^*, p_w b_w)] \geq \max(\mu^*, p^* b^* + (1 - p^*) p_w b_w)$$

Combining, we have

$$\max(V_{\sigma^1}, V_{\sigma^2}) \geq \max\left(\underbrace{p_w b_w + (1 - p_w)\mu^*}_{T1}, \underbrace{p^* b^* + (1 - p^*) p_w b_w}_{T2}, \underbrace{\mu^*}_{T3} \right) \tag{2.11}$$

Next, we complete the proof of the prophet inequality by showing that $\max(T1, T2, T3) \geq 0.8MAX$, where MAX was defined in (2.10) as the prophet's expected reward. The proof of this statement is largely algebraic and is provided in Appendix A.2. □

Tightness. Here is an example where the expected reward at the stopping time under the best ordering is arbitrarily close to $0.8 * E[\max_i X_i]$.

Example 1. *Let X_1 have support $\{0.5, \frac{1}{2\epsilon}\}$ with probabilities $\{1 - \epsilon, \epsilon\}$ for $\epsilon \in (0, 1)$, and X_2 have support $\{0, 1\}$ with probabilities $\{0.5, 0.5\}$. Then*

$$\begin{aligned} E[\max_i X_i] &= \epsilon \frac{1}{2\epsilon} + (1 - \epsilon)(1/2) * 1 + (1 - \epsilon)(1/2) * 0.5 \\ &= \frac{5}{4} - \frac{3}{4}\epsilon \end{aligned}$$

The two possible orderings (X_1, X_2) *and* (X_2, X_1) *are equally good here:*

$$V(X_1, X_2) = \epsilon\frac{1}{2\epsilon} + (1-\epsilon)0.5 = 1 - 0.5\epsilon$$

$$V(X_2, X_1) = 0.5 * 1 + 0.5 * (\epsilon\frac{1}{2\epsilon} + (1-\epsilon)0.5) = 1 - 0.5\epsilon$$

So in this example, the best ordering returns 0.8 *of the prophet's expected reward.*

2.5 Ordering three point distributions: FPTAS

Previously, in Section 2.3, we proved that ordering n random variables with three-point distributions is NP-hard even when each random variable X_i has a three-point distribution with support on $\{0, m_i, 1\}$. In this section, we provide an FPTAS for a slight generalization of this special case where the support is on three points $\{a_i, m_i, 1\}$ where $a_i \leq m_i \leq 1$. Note that in [25], the authors provide a PTAS for this problem with distributions that have support on any constant number points. The runtime complexity of the PTAS proposed there is $O(n^{2^\epsilon})$ in general. However, (to the best of our understanding) when applied to our problem, its complexity reduces to $O\left(\left(\frac{n}{\epsilon}\right)^{\epsilon^{-3}}\right)$. On other hand, for any $\epsilon \in (0, 1)$, our FPTAS runs in time $O(\frac{n^5}{\epsilon^2})$ (although it only applies to the above mentioned special case of 3-point support).

Theorem 4 (FPTAS for three-point distributions with same right endpoint)**.** *Given a set of n random variables* $X_1, \ldots, X_n$*, where each random variable* $X_i, i = 1, \ldots, n$ *has a three-point distribution with support on* $\{a_i, m_i, 1\}$ *for some* $m_i \in [0, 1]$ *and* $a_i \leq m_i$*. Then, there exists an algorithm that runs in time* $O(\frac{n^5}{\epsilon^2})$ *to find an ordering* σ *such that* $ALG = V_\sigma \geq (1-\epsilon)OPT$*. Here,* $OPT := V_{\sigma^*}$ *denotes the optimal expected reward at stopping time under an optimal ordering* σ^**.*

We first demonstrate an FPTAS for a special case where both left and right end points are the same for all i. Later we extend it to an FPTAS for the case when only the right end points are the same to prove Theorem 4.

Algorithm for same left and right end points. We are given a set of random variables $X_1, \ldots, X_n$ with three-point distributions (refer to Definition 2.2.3) where for each i, X_i takes values $\{0, m_i, 1\}$ with probabilities $1 - p_i - q_i$, p_i and q_i, respectively. The algorithm for finding an optimal ordering is based on the characterization of optimal orderings in this special case provided by Claim 2.3.1 in Section 2.3 (also see Remark 2.3.2).

Recall that in this case, there exists an optimal ordering σ that (under the optimal stopping policy) partitions the n variables into an ordered partition (S^σ, T^σ) (refer to Definition 2.3.3); and therefore the optimal ordering problem can be solved by finding an optimal ordered partition.

Specifically, define the collection $\mathcal{L}$ of ordered partitions as the collection of sequences (S, T) of n random variables that can be formed by partitioning the set of variables $\{X_1, \ldots X_n\}$ into two sets S and T with $T \neq \phi$, and then ordering the variables within S and within T in weakly descending order of E_i. Then, from Claim 2.3.1, we have that

$$\text{OPT} = \max_{(S,T)\in\mathcal{L}} V(S, T) \tag{2.12}$$

Using this observation, we designed an FPTAS to solve the problem in (2.12) of finding an optimal ordered partition. The idea behind the FPTAS is to discretize the interval $[0, 1]$ using a multiplicative grid with parameter $1 - \frac{\epsilon}{2n}$, so that it needs to search over only $poly(n, 1/\epsilon)$ partitions. A detailed description of the steps involved is provided below (Algorithm 4 and 5). Here, given a sequence A of random variables, $\{X_i, A\}$ denotes the sequence of random variables formed by concatenating a variable X_i to the beginning of the sequence A.

We prove the following theorem regarding Algorithm 4. The proof is in the appendix.

Theorem 5 (FPTAS for three-point distributions with same left and right endpoint)**.** *Given a set of n random variables $X_1, \ldots, X_n$, where each random variable $X_i, i = 1, \ldots, n$ has a three-point distribution with support on $\{0, m_i, 1\}$ for some $m_i \in [0, 1]$. Then, Algorithm 4, with parameters $\epsilon \in (0, 1)$ and $MAX = \frac{1}{2}OPT$ runs in time $O(\frac{n^4}{\epsilon^2})$ and finds an ordering σ such that $ALG = V_\sigma \geq (1 - \epsilon)OPT$. Here, $OPT := V_{\sigma^*}$ denotes the optimal expected reward at stopping time under an*

Algorithm 1 FPTAS for finding the optimal ordering through optimal partitioning

Input: Ordered sequence of variables $X_1, \ldots, X_n$ such that $E_1 \leq \cdots \leq E_n$, parameters MAX, ϵ.
Initialize: $\mathcal{L}^0 = \{(\phi, \phi)\}$, $\mathcal{L}^1 = \cdots = \mathcal{L}^n = \phi$;
for all $k = 1, \ldots, n$ **do**
  for all $(S, T) \in \mathcal{L}^{k-1}$ **do**
    Add two partitions $(\{X_k, S\}, T)$ and $(S, \{X_k, T\})$ to $\mathcal{L}^k$.
  end for
  Call Algorithm 5 to reduce the number of partitions in $\mathcal{L}^k$ by setting $\mathcal{L}^k \leftarrow \text{TRIM}(\mathcal{L}^k, \epsilon, \text{MAX})$.
end for
Return $\mathcal{L}^n$.

Algorithm 2 TRIM($\mathcal{L}, \epsilon, \text{MAX}$)

Initialize: $\rho := \left(1 - \frac{\epsilon}{2n}\right)$, $max := \max_{(S,T)\in\mathcal{L}} V(T)$, and $J := \max\{j : \rho^j max \geq \frac{\epsilon}{2n}\text{MAX}\}$.
Divide the partitions in $\mathcal{L}$ into $J + 1$ buckets as

$$\mathcal{B}_j := \{(S, T) : \rho^j max < V(T) \leq \rho^{j-1} max\}, \text{ for } j = 1, \ldots, J$$
$$\mathcal{B}_0 := \{(S, T) : T = \phi\}$$

Set $(S^j, T^j) := \arg\max_{(S,T)\in\mathcal{B}_j} V(S)$, for $j = 0, 1, \ldots, J$.
Return $\mathcal{L} := \{(S^j, T^j)\}_{j=0}^{J}$.

optimal ordering σ^*.

Now we are ready to prove Theorem 4 using an extension of Algorithm 4.

Proof. **Proof of Theorem 4:** Let $q_i := P(X_i = 1)$, $p_i := P(X_i = m_i)$ and so $P(X_i = 0) = 1 - p_i - q_i$. From Lemma 2.4.2, we know that there exists an optimal ordering σ with the property that $Pr(X_{\sigma(i)} \leq V_\sigma(i + 1)) > 0$ for $i = 1, \ldots, n - 1$. This implies that $a_{\sigma(i)} \leq V_\sigma(i + 1)$ and, as a consequence of the DP thresholds, if $X_{\sigma(i)} = a_{\sigma(i)}$ in this ordering, then the variable $X_{\sigma(i)}$ would be rejected. Define $X'_{\sigma(i)}$ to have support $\{0, m_{\sigma(i)}, 1\}$ and probabilities $\{1 - p_{\sigma(i)} - q_{\sigma(i)}, p_{\sigma(i)}, q_{\sigma(i)}\}$ for $i = 1, \ldots n - 1$, and $X'_{\sigma(n)} := E[X_{\sigma(n)}]$. Then, $V(X_{\sigma(1)}, \ldots, X_{\sigma(n)}) = V(X'_{\sigma(1)}, \ldots, X'_{\sigma(n)})$.

Now, if we knew $\sigma(n)$, then we could transform $X_{\sigma(i)}$ into $X'_{\sigma(i)}$ for $i = 1, \ldots, n$ and then, since $X'_{\sigma(i)}$ would have support on $\{0, m_i, 1\}$, we could use Algorithm 4 and Theorem 5. Since we do not know $\sigma(n)$, we run our FPTAS in Algorithm 4 n times. Here, in the i^{th} iteration, we define $X'_i := E[X_i]$ and X'_j to have support $\{0, m_j, 1\}$ and probabilities $\{1 - p_j - q_j, p_j, q_j\}$ for $j \neq i$.

And the algorithm is run on X'_j variables instead of X_js. In one of these iterations (specifically the iteration where $X'_i := E[X_i]$, for $i = \sigma(n)$) the ordering found by the algorithm will satisfy the required guarantees. □

Chapter 3: Optimal Ordering in the k-Secretary Problem

3.1 Introduction

In this chapter, we consider a *multiple-choices* extension to the Optimal Stopping problem in Chapter 2 where the difference is that the player is able to accept at most $k > 1$ rewards. Like in the previous setting, a player can probe a set of independent random variables $X_1, \ldots, X_n$ whose distributions are known to the player. After probing X_i, the player realizes its value, and has to decide whether to accept and earn reward X_i or to reject X_i. Once the player has accepted k random variables, the game stops. The goal of the player is to maximize the expected sum of all of the rewards chosen when the game finishes.

Similar to the $k = 1$ case, when the order in which the variables $X_1, \ldots, X_n$ are probed is fixed, the optimal stopping policy can be found using a dynamic program (DP). Suppose the order of observation is $(X_1, \ldots, X_n)$ and the player has finished examining variables $X_1, \ldots, X_{i-1}$ and can accept $k' > 0$ additional variables. Now, the player probes the next variable in the order, X_i, and its value is realized to the player. If the player accepts X_i, then he receives as a reward X_i plus the expected reward under an optimal policy from the remaining tail sequence $(X_{i+1}, \ldots, X_n)$ where he can accept $k' - 1$ more variables. However, if the player rejects X_i, he can still accept k' additional variables. Therefore, his expected reward is the reward under an optimal policy from $\{X_{i+1}, \ldots, X_n\}$ where he can accept k' additional variables. Whether or not the player accepts X_i depends on comparing the expected reward under these two alternatives and choosing the action with the greater expected reward.

Given that the optimal stopping policy under a fixed ordering is again easily determined by a DP, we focus our attention to the setting where the player can choose the order of observation. This is the *free order* setting, where the player not only has the power to choose the stopping policy, but

also on the order of observation. However, unlike the $k = 1$ case, which we examined in Chapter 2, there are two types of free order policies we need to distinguish: static (non-adaptive) policies and dynamic (adaptive) policies. In a static policy, the player decides on a *fixed* ordering of variables to examine a priori before examining any variables, and must follow this chosen ordering throughout the game. In a dynamic policy, there is no a priori fixed ordering and the player can decide on the next variable to examine based on the realizations and choices up to that point in time. In the $k = 1$ case, the optimal dynamic ordering is also static [36], hence this distinction disappears, but this distinction matters in the $k > 1$ case. The class of dynamic policies contains as a subset the class of static policies, hence an optimal dynamic policy always returns at least as much as an optimal static policy. Given this distinction, we may ask, what is the optimal static ordering and what is the optimal dynamic ordering? How much better is the optimal dynamic ordering over the optimal static ordering?

Prophet inequalities also extend to the $k > 1$ case, and they can be used to provide insight on the benefits of both static and dynamic orderings. It has a natural extension: The prophet's reward is defined as $E[\text{Sum of the largest k random variables}]$, and likewise can still be interpreted as the player's optimal choice in hindsight. A prophet inequality compares the expected reward of the player's stopping policy to the prophet's expected reward. To the best of our knowledge, the prophet inequalities so far studied in the $k > 1$ case are all in the adversarial ordering setting, where the player has no control over the order of observation. Hajiaghayi et al. [32] show that in this setting, there exists a strategy that can achieve a prophet inequality of $1 - O(\sqrt{\frac{\log(k)}{k}})$. Furthermore, they give a tight upper bound of $1 - O(\frac{1}{\sqrt{k}})$ based on a simple example. In their example, X_i is determinstally equal to 1 for $1 \leq i \leq \sqrt{k/8}$ (call this set A), takes values $\{0, 2\}$ with equal probability for $\sqrt{k/8} < i \leq 2k + \sqrt{k/8}$ (call this set B), and is determistically equal to 0 for the remaining variables. The main challenge in this example is that the player needs to determine how many variables to accept in set A, without knowing how many variables in set B realize 2. However, note that this example is trivial to solve in the static free order case - simply examine the random variables with support $\{0, 2\}$ first, accept any variable that realizes 2, and

if there are any spots remaining, examine the random variables that are deterministically equal to 1. This motivates the question of how much improvement we can get by choosing the order. Hajiaghayi et al.'s approximation factor was later improved to $1 - \frac{1}{\sqrt{k+3}}$ by Alaei [2], therefore giving a tight prophet inequality in the limit as k increases. A major focus of this chapter is to extend these prophet inequalities to the free order setting, for two-point distributions, in order to better understand the benefits of ordering.

In this chapter, we study the optimal ordering problem in the $k > 1$ setting when the distributions have small support. In the previous chapter, we showed that in the $k = 1$ setting, ordering two-point distributions is easy, but ordering three-point distributions is hard. In this chapter, we show that, in fact, find the static ordering of two-point distributions is NP-hard when $k = 2$. We then focus our study on two-point distributions. In particular, can we develop any algorithms to approximate the optimal ordering? Can we get prophet inequalities for either static or dynamic policies?

Our contributions. For negative results, we show that finding the optimal static ordering is NP-hard even in the $k = 2$ case. Even more, for the example we construct, the two-point distributions are very restricted, where all but one distribution has support on $\{0, 1\}$. This shows that even in very simple settings, the $k > 1$ setting is difficult to solve.

For positive results, we provide an FPTAS in the $k = 2$ case for two-point distributions. The idea of the FPTAS extends on the FPTAS we provided in Chapter 2. In addition, we provide two prophet inequalities for two-point distributions. For static policies, we provide an algorithm computing a static ordering with a $1 - O(\frac{\log(k)^{1/4}}{k^{3/4}})$ prophet inequality. For dynamic policies, we provide a $1 - 1/k$ prophet inequality with a corresponding example of tightness.

Organization of the chapter. The chapter is organized as follows. In Section 3.2, we provide preliminary notation and results that will be used throughout the rest of the chapter. In Section 3.3, we prove NP-hardness of finding the optimal static ordering under $k = 2$ for two-point distributions. In Section 3.4, we extend the FPTAS in Chapter 2 to ordering two-point distributions under $k = 2$. Finally, in Section 3.5, we prove prophet inequalities for two-point distributions. In 3.5.1,

we provide a prophet inequality for static order two-point distributions and in 3.5.2, we provide a prophet inequality for dynamic order two-point distributions.

3.1.1 Related Work

Most of the work that studies this problem in the $k > 1$ case assume an adversarial ordering setting. Notably, Hajiaghayi et al. [32] connected this problem to posted price mechanisms, where there are k identical objects and n buyers. They prove a $1 - O(\sqrt{\frac{\log(k)}{k}})$ prophet inequality by using a fixed threshold θ, which is set such that the expected number of variables larger than θ is $k - \sqrt{2k\log(k)}$, which is less than k. Then, they use concentration bounds to show that the event where the number of variables bigger than θ is between $k - 2\sqrt{2k\log(k)}$ and k occurs with high probability. Under this event happening, they will accept at least $(k - 2\sqrt{2k\log(k)})/k$ of the variables in the optimal offline, hence establishes their prophet inequality. We use many similar techniques in establishing our static prophet inequality in Section 3.5.1. They also give an example showing an upper bound of $1 - O(\frac{1}{\sqrt{k}})$ prophet inequality. Later Alaei [2] improved the prophet inequality to match this upper bound. Further work looked at the same adversarial setting but with matroid constraints (e.g. see [41], [24]).

In addition, the PTAS given by Fu et al. [25] mentioned in Chapter 2 also applies in the $k > 1$ dynamic setting (they call it the Committed ProbeTop-k Problem in their paper). For two-point distributions under $k = 2$, we improve on their result by giving an FPTAS for static orderings.

3.2 Preliminaries

Let $X_1, X_2, \ldots, X_n$ be random variables with distributions $D_1, D_2, \ldots, D_n$ respectively. An ordering σ is defined as a permutation of $\{1, \ldots, n\}$. Let $k > 0$ be a given constant, which is the number of rewards that the player can accept.

The Choose-k Secretary Problem under a fixed (static) ordering σ is defined as follows. At each time step $t = 1, \ldots, n$, suppose the player can accept k_t rewards remaining, where we initialize $k_1 = k$. The player then observes the value of $X_{\sigma(t)}$, generated from the distribution $D_{\sigma(t)}$. The

player can either accept $X_{\sigma(t)}$ or reject $X_{\sigma(t)}$. If the player rejects $X_{\sigma(t)}$, then the player proceeds to round $t+1$ with $k_{t+1} = k_t$. Otherwise, the player accepts $X_{\sigma(t)}$ reward, and if $k_t > 1$, the player proceeds to round $t+1$ with $k_{t+1} = k_t - 1$.

We define k stopping times $\tau_1, \ldots, \tau_k$. τ_i is defined as the time step t where the player accepts $X_{\sigma(t)}$ and have previously accepted $i-1$ variables. If the player has examined all of the variables and has only accepted $j < k$ variables, then we define $\tau_i = n+1$ with reward $X_{n+1} = 0$ for $i = j+1, \ldots, k$. The goal of the player is to maximize $E[\sum_{i=1}^{k} X_{\tau_i}]$, where the expectation is taken over both the stopping times τ_i, as well as the distributions X_{τ_i}, for $i = 1 \ldots, k$. We denote the total expected reward at the end of the game under the optimal stopping rule as $V_k(\sigma)$, so $V_k(\sigma) := E[\sum_{i=1}^{k} \tau_i^*]$, where τ_i^* are the stopping times given by the optimal stopping rule.

As in the $k = 1$ case, the optimal stopping rule for the $k > 1$ case under a fixed ordering σ can be defined by a dynamic program (DP). For a fixed ordering σ, we write $\sigma_i = (\sigma(i), \sigma(i+1), \ldots, \sigma(n))$ to denote the tail of the permutation σ (hence $\sigma_1 = \sigma$). Let $V_{k'}(\sigma_j)$ denote the optimal expected reward for the subsequence $X_{\sigma(j)}, \ldots, X_{\sigma(n)}$ when there are $k' \leq k$ choices remaining. Now, for any $k' \geq 1$ and $1 \leq j \leq n$:

$$V_{k'}(\sigma_j) = E[\max(X_{\sigma(j)} + V_{k'-1}(\sigma_{j+1}), V_{k'}(\sigma_{j+1}))] \tag{3.1}$$

$$V_{k'}(\sigma_{n+1}) = 0 \tag{3.2}$$

We can compute $V_{k'}(\sigma_j)$ for all $j = 1, \ldots, n$, $k' = 1, \ldots, k$, using backward induction. The optimal stopping policy can then be defined as follows: at any step t, if there are k' choices remaining, then accept $X_{\sigma(t)}$ if $X_{\sigma(t)} + V_{k'-1}(\sigma_{t+1}) \geq V_{k'}(\sigma_{t+1})$, otherwise reject. Therefore, $V_{k'}(\sigma_{t+1}) - V_{k'-1}(\sigma_{t+1})$ can be treated as a threshold of acceptance for $X_{\sigma(t)}$.

Let $S_1, S_2, \ldots, S_p$ be a partition of the indices $\{1, \ldots, n\}$. We sometimes use the notation $(S_1, S_2, \ldots, S_p)$ to denote a set of orderings σ where all of the indices in S_1 come first, followed by all of the indices in S_2, etc. We refer to $(S_1, S_2, \ldots, S_p)$ to be an *optimal partition* if there exists

$\sigma \in (S_1, S_2, \ldots, S_p)$ that is optimal. For simplicity, we sometimes abuse notation when a set S_i contains only a single random variable by just writing the random variable instead of the set. For example, if we have a partition S_1, S_2, S_3 and $S_2 = \{X_3\}$, then we may write (S_1, X_3, S_3) instead of (S_1, S_2, S_3).

We restrict our attention in this chapter to random variables that are *two-point* distributions, which were defined in 2.2.2. As we show, the problem of finding the optimal static ordering is NP-Hard in the very simple case of two-point distributions and $k = 2$, hence focusing on two-point distributions is already non-trivial. Since we only deal with two-point distributions, we will frequently write LEP, REP of a two-point variable X as a shorthand for its left and right endpoint (a and b), respectively. We will assume w.l.o.g. throughout this chapter that $X \geq 0$, since we will never accept X if $X < 0$.

For two-point distributions, there are only three choices that an optimal policy needs to consider when it reaches $X_{\sigma(i)}$:

1. Reject $X_{\sigma(i)}$
2. Accept $X_{\sigma(i)}$ iff its REP is realized
3. Accept $X_{\sigma(i)}$

Notice that an optimal policy would never accept the LEP and reject the REP. We can represent any such policy that only considers these three choices using a mapping $d : [n] \times [k'] \to \{0, 1, 2\}$, where $d(j, k') = 0$, $d(j, k') = 1$, and $d(j, k') = 2$ represents the three choices: Always reject X_j, accepts X_j iff $X_j = b_j$, and always accepts X_j, when we can accept k' more variables. We refer to d as a *decision function* and a pair (σ, d) represents a static policy. Given an order σ, it is straightforward to convert the optimal DP thresholds into a decision function d, and we will frequently say a decision function is *induced* from the DP thresholds (however we note that d may not be unique since there may be cases where accepting or rejecting a variable are equally optimal). We write $V_{k'}(\sigma, d)$ or $V_{k'}((X_{\sigma(1)}, \ldots, X_{\sigma(n)}), d)$, to denote the expectation of the policy (σ, d) when there are $k' \leq k$ choices remaining.

Throughout this chapter, we use the following lemma, which establishes that the DP threshold when the player can accept j variables is lower than the threshold when he can accept $j-1$ variables.

Lemma 3.2.1. *Let $X_1, \ldots, X_n$ be two-point distributions. For any ordering σ and $j \geq 2$,*

$$V_j(\sigma) - V_{j-1}(\sigma) \leq V_{j-1}(\sigma) - V_{j-2}(\sigma)$$

where $V_0(\sigma) := 0$.

Proof. We will fix $j \geq 2$ and assume w.l.o.g that $\sigma = (X_1, X_2, \ldots, X_n)$. We prove this lemma using induction on i over $\sigma_{n-i} = (X_{n-i}, X_{n-i+1}, \ldots, X_n)$, which is the tail of the sequence σ. The base case, $i = 0$ is true since $V_{j'}(\sigma_n) = E[X_n]$ for all $j' \geq 1$. Therefore, for $j \geq 3$, we have $V_j(\sigma_n) - V_{j-1}(\sigma_n) = V_{j-1}(\sigma_n) - V_{j-2}(\sigma_n) = 0$ and for $j = 2$, we have $V_j(\sigma_n) - V_{j-1}(\sigma_n) = 0 \leq V_{j-1}(\sigma_n) - V_{j-2}(\sigma_n) = E[X_n]$.

Now we prove the induction step. W.l.o.g. assume that the lemma is true for σ_{n-i+1} and we wish to prove it for σ_{n-i}. Define $p = P(X_{n-i} \geq V_j(\sigma_{n-i+1}) - V_{j-1}(\sigma_{n-i+1}))$ and $E = E[X_{n-i} | X_{n-i} \geq V_j(\sigma_{n-i+1}) - V_{j-1}(\sigma_{n-i+1})]$. Now, $V_j(\sigma_{n-i})$ can be written as:

$$V_j(\sigma_{n-i}) = p(E + V_{j-1}(\sigma_{n-i+1})) + (1-p)V_j(\sigma_{n-i+1})$$

In addition, we have

$$V_{j-1}(\sigma_{n-i}) \geq p(E + V_{j-2}(\sigma_{n-i+1})) + (1-p)V_{j-1}(\sigma_{n-i+1})$$

$V_{j-1}(\sigma_{n-i})$ is the expected reward under the optimal DP policy for σ_{n-i} when the player can accept $j-1$ more variables. Since this is optimal, it must upper bound the right hand side, which is the expected reward under the policy where we use the threshold $V_j(\sigma_{n-i+1}) - V_{j-1}(\sigma_{n-i+1})$ to determine whether or not we accept X_{n-i}. Hence, subtracting the two terms and using the induction hypothesis, we finish the induction:

$$
\begin{aligned}
V_j(\sigma_{n-i}) - V_{j-1}(\sigma_{n-i}) &\leq p(V_{j-1}(\sigma_{n-i+1}) - V_{j-2}(\sigma_{n-i+1})) + (1-p)(V_j(\sigma_{n-i+1}) - V_{j-1}(\sigma_{n-i+1})) \\
&\leq p(V_{j-1}(\sigma_{n-i+1}) - V_{j-2}(\sigma_{n-i+1})) + (1-p)(V_{j-1}(\sigma_{n-i+1}) - V_{j-2}(\sigma_{n-i+1})) \\
&= V_{j-1}(\sigma_{n-i+1}) - V_{j-2}(\sigma_{n-i+1}) \\
&\leq V_{j-1}(\sigma_{n-i}) - V_{j-2}(\sigma_{n-i})
\end{aligned}
$$

□

3.3 Hardness of Static orderings

In this section, we frequently use sets of $\{0, 1\}$ random variables. Let S be a set of $\{0, 1\}$ random variables. We use the notation $V_i(S)$ to denote the expected reward of the optimal policy where we accept a random variable if it realizes 1 and rejects otherwise, and we can accept up to i random variables. Notice that we can write this without ambiguity because the ordering within S does not matter.

We prove NP-hardness for finding the optimal static ordering when $k = 2$. The hardness result holds for instances that are surprisingly simple: All but one of the two-point distributions have support on $\{0, 1\}$. We prove hardness by reducing from the subset product problem.

Problem 2. ***Subset Product****: Given n positive integers $a_1, \ldots, a_n$ and a positive integer target A, does there exist a set U such that $\prod_{i \in U} a_i = A$?*

We will refer to our problem as the choose-2 static ordering problem:

Problem 3. ***Choose-2 Static Ordering****: Given random variables $X_1, \ldots, X_n$ and a constant c, does there exist an ordering σ such that $V_2(\sigma) \geq c$?*

Theorem 6. *The choose-2 static ordering problem is NP-Hard.*

Proof. We will show NP-hardness using a reduction from subset product. Let $a_1, \ldots, a_n$ and A be an instance of subset product. We will construct an instance of the choose-2 static ordering

problem and show that subset product is satisfiable iff $V_2(\sigma) \geq c$ for some ordering σ (of the constructed random variables) and a constant c (to be set later).

Let X_i be random variables with support $\{0, 1\}$ and probabilities $\{\frac{1}{a_i}, 1 - \frac{1}{a_i}\}$, $i = 1, \ldots, n$. Let X_{n+1} be a random variable with support $\{0, 1\}$ and probabilities $\{\frac{1}{a_{n+1}}, 1 - \frac{1}{a_{n+1}^2}\}$, where $a_{n+1} := A^2$. Note that these probabilities are all well defined since $a_i \geq 1$ and $A \geq 1$. Define $\gamma = \prod_{i=1}^{n+1} \frac{1}{a_i}$, which is the probability that all of $X_1, \ldots, X_{n+1}$ realize 0. Finally, we define a random variable X_{n+2} with support $\{1 - \gamma A^2, r\}$ and probabilities $\{1 - p, p\}$, where we define r and p such that $E[X_{n+2}] = pr + (1 - p)(1 - \gamma A^2) > 1$ (any such r and p is fine).

Let S and T define a partition of $\{X_1, \ldots, X_n, X_{n+1}\}$, and define $\gamma_S = \prod_{i \in S} \frac{1}{a_i}$ and $\gamma_T = \frac{\gamma}{\gamma_S}$. γ_S (γ_T) is the probability that every element in S (T) realizes 0. S and T will be used to define the set of random variables examined before and after X_{n+2}, respectively.

The next few lemmas help characterize the set of optimal stopping policies.

Lemma 3.3.1. *Wlog assume that (S, X_{n+2}, T) is an optimal partition and let $\sigma \in (S, X_{n+2}, T)$ be an optimal ordering. There exists an optimal policy (σ, d) where d is a decision function induced from the DP thresholds (defined in Section 3.2), that satisfies the following properties:*

1. $d[n + 2, 1] \in \{1, 2\}$.
2. $d[n + 2, 2] = 2$.
3. *If $X_i \in S$, then $d[i, 2] = 1$.*
4. *If $X_i \in S$, then $d[i, 1] = 0$.*
5. *If $X_i \in T$, then $d[i, 1] = 1$.*

Proof. Since d is induced from DP thresholds, recall that the threshold for acceptance for $X_{\sigma(i)}$ is $V_2(\sigma_{i+1}) - V_1(\sigma_{i+1})$ when we can accept two more variables and $V_1(\sigma_{i+1})$ when we can accept one more variable. If $X_{\sigma(i)}$ is strictly larger (smaller) than this threshold, it is strictly better to accept (reject).

1. When the player can accept one more variable, $V_1(T)$ is the threshold for X_{n+2}. This then follows since $r > 1 > V_1(T)$, hence it is optimal to accept X_{n+2} when $X_{n+2} = r$.

2. $d[n+2, 2] \geq 1$ follows from $r > 1 > V_1(T) \geq V_2(T) - V_1(T)$ where the last inequality is due to Lemma 3.2.1. We now show that the LEP of X_{n+2} is also accepted, hence $d[n+2, 2] = 2$. Let γ_T^1, γ^1, γ', and γ'^1 be the probabilities that exactly one random variable in T realizes 1, that exactly one element in $\{X_1, \ldots, X_{n+1}\}$ realizes 1, that no element in $\{X_1, \ldots, X_n\}$ realizes 1 (hence $\gamma = \frac{\gamma'}{A^2}$), and that exactly one element in $\{X_1, \ldots, X_n\}$ realizes 1, respectively. Then, the threshold is

$$\begin{aligned}
V_2(T) - V_1(T) &= [2(1 - \gamma_T - \gamma'_T) + \gamma_T^1] - [1 - \gamma_T] \\
&= [2(1 - \gamma_T - \gamma_T^1) + \gamma_T^1] - [\gamma_T^1 + (1 - \gamma_T - \gamma_T^1)] \\
&= 1 - \gamma_T - \gamma_T^1 \\
&\leq 1 - \gamma - \gamma^1 \quad \text{(Since } T \subseteq \{X_1, \ldots, X_{n+1}\}\text{)} \\
&= 1 - \frac{\gamma'}{A^2} - [\frac{\gamma'^1}{A^2} + (1 - \frac{1}{A^2})\gamma'] \\
&= 1 - \gamma' - \frac{\gamma'^1}{A^2}
\end{aligned}$$

   Now, $1 - \gamma A^2 = 1 - \gamma' > 1 - \gamma' - \frac{\gamma'^1}{A^2}$, and the result follows since $\gamma'^1 > 0$.

3. Suppose we can accept two more variables. If it is optimal to reject X_i, then it is also optimal to reject all remaining variables in S. This is because if there is an optimal policy that rejects X_i and accepts some $X_j \in S$ where $j > i$, then the same policy that accepts X_i and rejects X_j must also be optimal.

   Due to 1 and 2, the expectation of this policy is $E[X_{n+2}] + V_1(T)$. Now, $V_1(T) < 1$, hence the policy that accepts X_i and X_{n+2} returns $1 + E[X_{n+2}]$, a contradiction to optimality. Hence, it is better to accept X_i, so $d[i, 2] = 1$.

4. Since $X_i \in S$, X_{n+2} has yet to be examined. Since, $1 < E[X_{n+2}]$, it is optimal to reject X_i, so $d[i, 1] = 0$.

5. Because of 1 and 2, once we reach the variables in T, we can only accept one more variable. If we reject $X_i = 1$, then the expected reward is the probability that a 1 will be realized in the remaining variables, which is less than 1. Hence $d[i, 1] = 1$.

□

By construction, under an optimal policy (σ, d) defined in this lemma, we can never reach a variable in T and still need to accept two more variables, because of 2 in Lemma 3.3.1. The only ambiguity that remains is whether $d[n + 2, 1] = 1$ or $d[n + 2, 1] = 2$. We define Ω to be the class of optimal policies (σ, d) that satisfies Lemma 3.3.1, and $d[n + 2, 1] = 1$. We define Ω' to be the optimal policies that satisfy Lemma 3.3.1 and $d[n + 2, 1] = 2$.

The next lemma shows that any optimal policy in Ω' is "dominated" by an optimal policy in Ω.

Lemma 3.3.2. *Suppose there exists an optimal policy in Ω' with an ordering in (S, X_{n+2}, T). Then, there exists a policy in Ω with an ordering in (X_{n+2}, S, T) that is also optimal.*

Proof. Let (σ, d) be the optimal policy in Ω' with an ordering in (S, X_{n+2}, T). By number 4 in Lemma 3.3.1, (σ, d) cannot accept two variables in S. By the definition of Ω', this implies that any policy in Ω' always examines and accepts X_{n+2}. Hence, this policy returns no more than $E[X_{n+2}] + V_1(S \cup T)$.

Now consider the partition (S', X_{n+2}, T') where $S' = \emptyset$ and $T' = S \cup T$. Consider the policy that accepts X_{n+2} regardless and then accepts a variable in T' iff its REP is realized. This policy returns $E[X_{n+2}] + V_1(S \cup T)$. Since X_{n+2} is the first variable, this policy is in Ω by default and it is also optimal. □

Because of Lemma 3.3.2, we know that there exists an optimal policy in Ω. The intuition for the remainder of the proof is to show that subset product is satisfiable iff there exists a policy in Ω such that $\gamma_S = \frac{1}{A}$. There is a natural tradeoff between having a large (small) γ_S and a small (large)

γ_T. If we have a large γ_S, then this means there is a high probability that all variables in S realize 0, which means that X_{n+2} must be accepted even if $X_{n+2} = 1 - \gamma A^2$. This ordering underperforms if there are two random variables in T that realize 1. On the other hand, if γ_S is small, it is likely that more than one random variable in S realizes 1. However, due to number 4 in Lemma 3.3.1, only one of these random variables is accepted and the rest are "wasted" because it is optimal to always examine X_{n+2}. This tradeoff implies that γ_S is maximized at a particular value, and this instance is constructed such that the maximum occurs when $\gamma_S = \frac{1}{A}$. The next lemma provides an explicit formulation of the expected reward for any policy in Ω.

Lemma 3.3.3. *Given S and T, the expected reward of the policy in Ω is:*

$$f(\gamma_S) := [1 + pr + (1-p)(1+\gamma) - \gamma] - (1-p)[\gamma\gamma_S A^2 + \frac{\gamma}{\gamma_S}]$$

Furthermore, f is strictly concave and attains a unique maximum at $\frac{1}{A}$.

Proof. The expected return can be written as

$$\begin{aligned}
f(\gamma_S) &= (1-\gamma_S)[1 + pr + (1-p)(1-\gamma_T)] + \gamma_S[pr + (1-p)(1-\gamma A^2) + (1-\gamma_T)] \\
&= 1 + pr - \gamma_S - pr\gamma_S + (1-p)(1-\gamma_S)(1 - \frac{\gamma}{\gamma_S}) + pr\gamma_S + (1-p)(1-\gamma A^2)\gamma_S + \gamma_S(1 - \frac{\gamma}{\gamma_S}) \\
&= 1 + pr - \gamma_S + (1-p)(1 - \gamma_S - \frac{\gamma}{\gamma_S} + \gamma) + (1-p)(1-\gamma A^2)\gamma_S + \gamma_S - \gamma \\
&= [1 + pr + (1-p)(1+\gamma) - \gamma] - (1-p)[(1 - (1-\gamma A^2))\gamma_S + \frac{\gamma}{\gamma_S}]
\end{aligned}$$

Taking the derivative, we find that:

$$f'(\gamma_S) = -(1-p)\gamma A^2 + \frac{(1-p)\gamma}{\gamma_S^2}$$

Setting it equal to zero and solving, we find that the root occurs at $\gamma_S = \frac{1}{A}$. Taking the second derivative proves that f is strictly concave:

$$f''(\gamma_S) = -\frac{2(1-p)\gamma}{\gamma_S^3} < 0$$

□

Now we finish the proof of Theorem 1 by showing that subset product is satisfiable iff there exists an optimal policy in Ω such that $\gamma_S = \frac{1}{A}$ and no optimal policy exists outside of Ω.

Suppose subset product is satisfiable and is satisfied by a subset S. Taking the corresponding S for our choose-2 static ordering instance, we get that $\gamma_S = \frac{1}{A}$, hence, by Lemma 3.3.3, there is no policy in Ω that is better than this policy. Furthermore, by Lemma 3.3.2, there is no optimal policy in Ω' since any such policy will have the same expected reward as the policy in Ω with order (X_{n+2}, S, T). The reward of this policy is $f(0)$. However, since f is strictly concave, $f(0) < f(1/A)$ and is not optimal.

Now suppose there is an optimal policy in Ω such that $\gamma_S = \frac{1}{A}$. The corresponding set S in the subset product instance satisfies $\prod_{i \in S} a_i = A$, hence subset product is satisfiable. □

3.4 FPTAS for Two-Point Static Orderings

In the previous section, we proved that finding the optimal static ordering is NP-Hard even for the $k = 2$ case. In this section, we provide an FPTAS for approximating the optimal static ordering for two-point distributions in the $k = 2$ case. For this section, we define OPT $= \max_\sigma V_2(\sigma)$, which is the expected reward under the optimal static policy for the order σ. In addition, we assume w.l.o.g. that all support points are distinct in this section.

We will prove the following theorem in this section:

Theorem 7 (FPTAS for Choose-2 Two-Point Distributions)**.** *Let $X_1, \ldots, X_n$ be two-point distributions. Algorithm 3 runs in time $O(\frac{n^8}{\epsilon^5})$ and finds an ordering σ such that ALG $= V_2(\sigma) \geq (1-\epsilon)OPT$.*

First, we define a subset of policies which we call P'. We will prove that P' contains an optimal policy and the FPTAS we design will only be searching through policies in P'.

Definition 3.4.1. *Define a set of policies $P' \subset P$ such that if $(\sigma, d) \in P'$, then $\sigma \in (S, Y, T, Z)$ for some S, Y, T, Z, where Y, Z are random variables, and S, T are sets of random variables. Furthermore (σ, d) satisfies the following properties for all i, j:*

1. *If $X_i = Y$, then $d[i, 2] = 2$.*
2. *If $X_i \in S$, then $d[i, 2] = 1$*
3. *$d[i, 1]$ is induced from DP thresholds.*
4. *If $X_i, X_j \in S$ and $i < j$, then $b_i > b_j$ (i.e. S is ordered by decreasing REP).*
5. *If $X_i, X_j \in T$ and $i < j$, then $b_i > b_j$ (i.e. T is ordered by decreasing REP).*

Now, we define some notation that we use throughout this section. For a given partition (S, Y, T, Z), from our assumption that support points are distinct, there is a unique ordering σ that satisfies definition 3.4.1. Therefore, we may write (S, Y, T, Z) as a substitute for σ and hence may write $V_i(S, Y, T, Z)$ to mean $V_i(\sigma)$ for $i = 1, 2$. If S and Y are empty, we may write $V_1(T, Z)$ instead of $V_1(S, Y, T, Z)$.

There may be multiple decision functions d such that $(\sigma, d) \in P'$ because of 3 (the optimal DP decision may be indifferent to accepting or rejecting a reward), however they all have the same expected reward. We will write $V'_i(S, Y, T, Z)$ to denote the expected reward of the policy $(\sigma, d) \in P'$ where $\sigma \in (S, Y, T, Z)$ when there are i choices remaining. Note that, because of 3 in Definition 3.4.1, $V'_1(S, Y, T, Z) = V_1(S, Y, T, Z)$, i.e. the expected reward of the policy in P' is optimal.

Finally, for any constant c, we sometimes write $V_1(S, Y, c)$ to mean $V_1(\sigma')$ where σ' orders everything in S in decreasing REP, then Y, then c, where we treat c as a degenerate (constant) random variable.

Lemma 3.4.2. *There exists a policy $(\sigma, d) \in P'$ that is optimal.*

Proof. Let $(\sigma^*, d^*) \in P$ be an optimal policy, where d^* is induced from DP thresholds. Let $j = \min\{i : d^*[\sigma^*(i), 2] = 2\}$ and define $Y := X_{\sigma^*(j)}$, and $S = \{X_{\sigma^*(1)}, \ldots, X_{\sigma^*(j-1)}\}$. By the definition of j, $d^*[\sigma^*(i), 2] \leq 1$ for all $X_{\sigma^*(i)} \in S$. Y is the first variable we accept regardless when there are two choices remaining, and S are all of the variables ordered before Y.

Now, suppose $d^*[\sigma^*(i), 2] = 0$ for some $X_{\sigma^*(i)} \in S$. Then, by Lemma 3.2.1, $d^*[\sigma^*(i), 1] = 0$, and hence $X_{\sigma^*(i)}$ is always rejected. Since $X_{\sigma^*(i)}$ is always rejected, it does not matter when the policy examines $X_{\sigma^*(i)}$, so we can assume that $X_{\sigma^*(i)}$ is examined after Y and the ordering would still be optimal. Therefore, we can assume wlog that $d^*[\sigma^*(i), 2] = 1$. Also, notice that once we reach $X_{\sigma^*(j+1)}$, the player can accept at most one more variable, since $d^*[\sigma^*(j), 2] = 2$. Therefore, we can assume that $\{X_{\sigma^*(j+1)}, \ldots, X_{\sigma^*(n)}\}$ is ordered using an optimal $k = 1$ algorithm. From Proposition 2.4.4, we can assume that $T := \{X_{\sigma^*(j+1)}, \ldots, X_{\sigma^*(n-1)}\}$ are ordered in decreasing REP. Define $Z := X_{\sigma^*(n)}$.

It remains to show that S can be optimally ordered in decreasing REP. We prove this using an interchange argument. Suppose we do not examine S by decreasing REP, so there are two adjacent variables $X_{\sigma^*(i)}$ and $X_{\sigma^*(i+1)}$ in S such that $b_{\sigma^*(i)} < b_{\sigma^*(i+1)}$. Define the ordering $\sigma' := (\sigma^*(1), \ldots, \sigma^*(i-1), \sigma^*(i+1), \sigma^*(i), \ldots, \sigma^*(n))$, where $\sigma^*(i), \sigma^*(i+1)$ are swapped and the rest of the ordering is intact; w.l.o.g. assume i is the largest index in S for which this is the case. By induction, it is enough to show that $V_l((\sigma^*(i+1), \sigma^*(i), \ldots, \sigma^*(n)), d^*) \geq V_l((\sigma^*(i), \sigma^*(i+1), \ldots, \sigma^*(n)), d^*)$ for $l = 1, 2$, which would imply that $V_2(\sigma', d^*) \geq V_2(\sigma^*, d^*)$. This is true for $l = 1$ using a standard interchange argument for two point distributions.

We prove this for $l = 2$. Since $X_{\sigma^*(i)}, X_{\sigma^*(i+1)} \in S$, $d^*[\sigma^*(i), 2] = d^*[\sigma^*(i+1), 2] = 1$. Because d^* is induced from DP thresholds on the ordering σ^*, using Lemma 3.2.1, $d^*[\sigma^*(i), 1] \leq 1$ and $d^*[\sigma^*(i+1), 1] \leq 1$, hence the LEPs of $X_{\sigma^*(i)}, X_{\sigma^*(i+1)}$ are always rejected if realized. Notice that if either $X_{\sigma^*(i)}$ or $X_{\sigma^*(i+1)}$ realize their LEP, then (σ', d^*) and (σ^*, d^*) return the same value. Therefore, we can assume that both variables realize their REP and we will show that the policy (σ', d^*) always returns at least as much as (σ^*, d^*). Let $C := V_1((X_{\sigma^*(i+2)}, \ldots, X_{\sigma^*(n)}), d^*)$. We consider the following cases:

1. $b_{\sigma^*(i)} < b_{\sigma^*(i+1)} < C$. In this case, (σ', d^*) examines $X_{\sigma^*(i+1)}$ first and accepts it. d^* then rejects $X_{\sigma^*(i)}$ since $b_{\sigma^*(i)} < C$. In contrast, (σ^*, d^*) examines $X_{\sigma^*(i)}$ first and accepts it, but rejects $X_{\sigma^*(i+1)}$. Both policies accept the same second reward. Hence, (σ', d^*) returns more than (σ^*, d^*).

2. $b_{\sigma^*(i)} < C < b_{\sigma^*(i+1)}$. In this case, (σ', d^*) accepts $X_{\sigma^*(i+1)}$ and rejects $X_{\sigma^*(i)}$, giving an expected return of $b_{\sigma^*(i+1)} + C$. In contrast, (σ^*, d^*) examines $X_{\sigma^*(i)}$ first and accepts, then examines $X_{\sigma^*(i+1)}$ and also accepts since $C < b_{\sigma^*(i+1)}$. Therefore, (σ^*, d^*) returns $b_{\sigma^*(i)} + b_{\sigma^*(i+1)}$. Since $b_{\sigma^*(i)} < C$, (σ', d^*) returns on expectation more than (σ^*, d^*).

3. $C < b_{\sigma^*(i)} < b_{\sigma^*(i+1)}$. In this case, both policies accept both variables and hence returns $b_{\sigma^*(i)} + b_{\sigma^*(i+1)}$.

In all three cases, (σ', d^*) returns at least as much as (σ^*, d^*), hence must also be optimal. Since (σ', d^*) is optimal, d^* must be a DP decision function for σ'. By induction, we can repeatedly use the same interchange argument, to order S by decreasing REP and the resulting order will be optimal.

Let σ'' be the final ordering. It is clear that $\sigma'' \in (S, Y, T, Z)$ and, by the above argument, S and T are examined in decreasing REP. In addition, (σ', d^*) is optimal, and as we argued, $d^*[i, 2] = 2$ for $X_i = Y$ and $d^*[i, 2] = 1$ for $X_i \in S$. Therefore, $(\sigma', d^*) \in P'$. □

Due to this lemma, we will now restrict our attention to policies in P'. If X_1 is the random variable in S with the largest REP, note that we can write

$$V_2'(S, Y, T, Z) = p_1(b_1 + V_1(S \backslash X_1, Y, T, Z)) + (1 - p_1)V_2'(S \backslash X_1, Y, T, Z) \tag{3.3}$$

using the fact that $d[1, 2] = 1$. Also, we have

$$V_2'(Y, T, Z) = E[Y] + V_1(T, Z) \tag{3.4}$$

using the fact that Y is accepted regardless when the player can accept two more variables.

Now, the FPTAS is designed to return a policy in P' that approximates OPT. Let (S^*, Y^*, T^*, Z^*) be defined such that OPT $= V_2'(S^*, Y^*, T^*, Z^*)$. The algorithm first fixes two variables to be Y and Z and also takes a "guess" at $V_1(T^*, Z^*)$, which we denote as V^*. It iteratively considers the remaining variables $\{X_1, \ldots, X_n\} \backslash \{Y \cup Z\}$ in order of increasing right endpoint, and partitions them into either S or T. At iteration i, the algorithm maintains a set of partitions (S_{i1}, Y, T_{i1}, Z), (S_{i2}, Y, T_{i2}, Z), ..., where S_{ij}, T_{ij} is a partition of the i variables in $\{X_1, \ldots, X_n\} \backslash \{Y \cup Z\}$ with the smallest REP.

During iteration i, the algorithm builds from $\{(S_{i-1,1}, Y, T_{i-1,1}, Z), (S_{i-1,2}, Y, T_{i-1,2}, Z), \ldots\}$ to get new partitions $\{(S_{i1}, Y, T_{i1}, Z), (S_{i2}, Y, T_{i2}, Z), \ldots\}$. For each partition (S_{ij}, Y, T_{ij}, Z), we record a pair $(V_1(T_{ij}, Z), V_1(S_{ij}, Y, V^*))$. If we have two pairs $(V_1(T_{ij}, Z), V_1(S_{ij}, Y, V^*))$, $(V_1(T_{ij'}, Z), V_1(S_{ij'}, Y, V^*))$, for which $V_1(T_{ij}, Z) \approx V_1(T_{ij'}, Z)$ and $V_1(S_{ij}, Y, V^*) \approx V_1(S_{ij'}, Y, V^*)$, we keep one pair (based on whether $V_2'(S_{ij}, Y, V^*)$ or $V_2'(S_{ij'}, Y, V^*)$ is larger) and discard the other. The algorithm continues until all random variables in $\{X_1, \ldots, X_n\} \backslash \{Y \cup Z\}$ are partitioned into either S or T.

At the end of all iterations, we have a set of partitions $\{(S_1, , Y, T_1, Z), (S_2, Y, T_2, Z), \ldots\}$. Assuming that $V^* \approx V_1(T^*, Z^*)$, there exists random variables Y', Z', such that the algorithm applied to V^*, Y', Z' returns a partition (S_j, Y', T_j, Z') satisfying $V_1(T_j, Z') \approx V_1(T^*, Z^*)$ and $V_1(S_j, Y', V^*) \approx V_1(S^*, Y^*, V^*) \approx V_1(S^*, Y^*, T^*, Z^*)$. Since we do not know which variables satisfy the roles of Y', Z', we loop through all pairs and take different guesses for V^*.

The full algorithm is described in Algorithm 3. Here, given a sequence A of random variables, $\{X_i, A\}$ denotes the sequence of random variables formed by concatenating a variable X_i to the beginning of sequence A.

Now, we prove some lemmas that will help in our analysis.

Lemma 3.4.3. *Assume $\sigma = (X_1, \ldots, X_n)$ and let (σ, d) be a policy where $d[n, 1] = d[n, 2] = 2$. Let $\sigma' = (X_1, \ldots, X_{n-1})$. Then,*

$$V_2(\sigma', d) \geq V_2(\sigma, d) - E[X_n]$$

Algorithm 3 MASTER FPTAS

Input: Random variables $X_1, \ldots, X_n$, parameters MAX, ϵ.
Initialize $\omega = (1 - \frac{\epsilon}{3})$. $J' = \lceil \log(\frac{\epsilon}{3n}) / \log(1 - \frac{\epsilon}{3}) \rceil$. $\mathcal{L}'' = \emptyset$
for all $i = 1, \ldots, n$ **do**
  for all $j = 1, \ldots, i-1, i+1, \ldots, n$ **do**
    if $E[X_j] \geq \frac{\epsilon}{3n}$MAX **then**
      for all $\ell = 0, 1 \ldots J'$ **do**
        $\mathcal{L}'' = \mathcal{L}'' \cup \text{FPTAS}(X_i, X_j, 2\text{MAX}\omega^\ell)$
      end for
    end if
  end for
end for
Return $\max_{(S,Y,T,Z) \in \mathcal{L}''} V_2'(S, Y, T, Z)$

Algorithm 4 FPTAS(Y, Z, V)

Input: Y, Z, and ordered sequence of remaining variables $X_1, \ldots, X_{n-2}$ such that $b_1 \leq \cdots \leq b_{n-2}$, parameters MAX, ϵ, V.
Initialize: $\mathcal{L}^0 = \{(\phi, Y, \phi, Z)\}$, $\mathcal{L}^1 = \cdots = \mathcal{L}^{n-2} = \phi$;
for all $k = 1, \ldots, n-2$ **do**
  for all $(S, Y, T, Z) \in \mathcal{L}^{k-1}$ **do**
    Add two partitions $(\{X_k, S\}, Y, T, Z)$ and $(S, Y, \{X_k, T\}, Z)$ to $\mathcal{L}^k$.
  end for
  Call Algorithm 5 to reduce the number of partitions in $\mathcal{L}^k$ by setting $\mathcal{L}^k \leftarrow \text{TRIM}(\mathcal{L}^k, \epsilon, \text{MAX}, V)$.
end for
Return $\mathcal{L}^{n-2}$

Proof. Let A be the event that X_n is not examined in the policy (σ, d) and let $p = P(A)$. Let $V_2(\sigma, d|A)$ be the expected reward of the policy (σ, d) conditioned on event A occurring. Conditioned on A occurring, $V_2(\sigma', d|A) = V_2(\sigma, d|A)$ and conditioned on A not occurring (so X_n is examined, and since it is the last variable, is accepted), $V_2((X_1, \ldots, X_{n-1}), d|\text{not A}) = V_2((X_1, \ldots, X_n), d|\text{not A}) - E[X_n]$. Hence,

Algorithm 5 TRIM($\mathcal{L}, \epsilon, \text{MAX}, V$)

Initialize: $\rho := \left(1 - \frac{\epsilon}{6n}\right)$,
$max_1 := \max_{(S,Y,T,Z)\in\mathcal{L}} V_1(T, Z)$,
$max_2 := \max_{(S,Y,T,Z)\in\mathcal{L}} V_1(S, Y, V)$
$J_1 := \max\{j : \rho^j max_1 \geq \frac{\epsilon}{3n}\text{MAX}\}$,
$J_2 := \max\{j : \rho^j max_2 \geq \frac{\epsilon}{3n}\text{MAX}\}$,
Divide the partitions in $\mathcal{L}$ into $(J_1 + 1)(J_2 + 1)$ buckets as

$$\mathcal{B}_{j_1,j_2} := \{(S, Y, T, Z) : \rho^{j_1} max_1 < V_1(T, Z) \leq \rho^{j_1-1} max_1, \rho^{j_2} max_2 < V_1(S, Y, V) \leq \rho^{j_2-1} max_2\}, \text{ for } j_1 = 1, \ldots, J_1 + 1, j_2 = 1, \ldots, J_2 + 1$$

Set $(S^{j_2}, Y, T^{j_1}, Z) := \arg\max_{(S,T)\in\mathcal{B}_{j_1,j_2}} V_2'(S, Y, V)$, for $j_1 = 0, 1, \ldots, J_1 + 1$, $j_2 = 0, 1, \ldots, J_2 + 1$.
Return $\mathcal{L} := \{(S^j, Y, T^j, Z)\}_{(j_1,j_2)=(0,0)}^{(J_1+1,J_2+1)}$.

$$\begin{aligned}
V_2((X_1, \ldots, X_{n-1}), d) &= (1 - p)V_2((X_1, \ldots, X_{n-1}), d|\text{not A}) + pV_2((X_1, \ldots, X_{n-1}), d|A) \\
&= (1 - p)V_2((X_1, \ldots, X_n), d|\text{not A}) + p(V_2((X_1, \ldots, X_n), d|\text{A}) - E[X_n]) \\
&= V_2((X_1, \ldots, X_n), d) - pE[X_n] \\
&\geq V_2((X_1, \ldots, X_n), d) - E[X_n]
\end{aligned}$$

□

Lemma 3.4.4. *Given* $0 \leq \epsilon \leq 1$, *let* $P'' \subset P'$ *be the set of all policies* (σ, d) *such that* $E[X_{\sigma(n)}] \geq \frac{\epsilon OPT}{3n}$. *Let* $OPT' = \max_{(\sigma,d)\in P''} V_2(\sigma, d)$. *Then,*

$$OPT' \geq (1 - \frac{\epsilon}{3})OPT$$

Proof. By Lemma 3.4.2, there exists an optimal policy (σ^*, d^*) in P', hence $\sigma^* \in (S^*, Y^*, T^*, Z^*)$ for some S^*, Y^*, T^*, Z^*. W.l.o.g. assume that $\sigma^* = (X_1, \ldots, X_n)$. By definition, $X_n = Z^*$ and suppose $X_l = Y^*$ for some l. Assume first that there exists some j such that $E[X_i] < \frac{\epsilon\text{OPT}}{3n}$ for all $j < i \leq n$, and $E[X_j] \geq \frac{\epsilon\text{OPT}}{3n}$. Then, by repeated application of Lemma 3.4.3:

$$
\begin{aligned}
V_2((X_1, \ldots, X_j), d^*) &\geq V_2((X_1, \ldots, X_n), d^*) - \sum_{i=j+1}^{n} E[X_i] \\
&= \text{OPT} - (n-j)\frac{\epsilon \text{OPT}}{3n} \\
&\geq (1 - \frac{\epsilon}{3})\text{OPT}
\end{aligned}
$$

Define $Z' = X_j$. Suppose first that $Y^* \in \{X_1, \ldots, X_j\}$, and define $Y' = Y^*$. Define $S' = S^*$ and T' be the remaining elements in $\{X_1, \ldots, X_j\}$. Define the decision function d' to be identical to d^*, except $d'[j, 1] = d'[j, 2] = 2$ (this is optimal when X_j is the last variable). Then, $V_2((X_1, \ldots, X_j), d') \geq V_2((X_1, \ldots, X_j), d^*) \geq (1 - \frac{\epsilon}{3})\text{OPT}$.

If $Y^* \notin \{X_1, \ldots, X_j\}$, then define $Y' = X_{j-1}$. Define $S' = \{X_1, \ldots, X_{j-2}\}$ and $T' = \emptyset$. Define d' to be identical to d^* except $d'[j-1, 2] = 2$ and $d'[j, 1] = d'[j, 2] = 2$ (this is optimal when X_{j-1} is the second to last variable and X_j is the last variable). Then, $V_2((X_1, \ldots, X_j), d') \geq V_2((X_1, \ldots, X_j), d^*) \geq (1 - \frac{\epsilon}{3})\text{OPT}$.

For both of these cases, we can add the variables $\{X_{j+1}, \ldots, X_n\}$ to the set T' to get a partition (S', Y', T', Z') on all the variables. Let σ' be the order that examines S' and T' in decreasing REP. We can define d' to follow the optimal DP thresholds on the variables in T'. We can easily see that $(\sigma', d') \in P'$. Because d' follows the DP thresholds, adding $\{X_{j+1}, \ldots, X_n\}$ to T' can only increase the expected reward. Therefore, $V_2(\sigma', d') \geq V_2((X_1, \ldots, X_j), d') \geq (1 - \frac{\epsilon}{3})\text{OPT}$.

Now, if no such j exists, then by Lemma 3.4.3,

$$
\begin{aligned}
0 &\geq V_2((X_1, \ldots, X_n), d^*) - \sum_{i=1}^{n} E[X_i] \\
&\geq (1 - \frac{\epsilon}{3})\text{OPT}
\end{aligned}
$$

This implies that OPT = 0, a contradiction. □

Using Lemma 3.4.4, we need to only consider orderings where the expectation of the last element is not "small," which lets us bound $|\{\mathcal{B}_{j_1,j_2}\}_{j_1,j_2}|$ in Algorithm 5.

Lemma 3.4.5. *Let $\epsilon \in (0,1)$. Suppose (σ^*, d^*) is an optimal policy in P'' (as defined in Lemma 3.4.4) and suppose (S^*, Y^*, T^*, Z^*) is the corresponding ordered partition (hence $E[Z^*] \geq \frac{\epsilon OPT}{3n}$). Let $OPT' = \max_{(\sigma,d)\in P''} V_2(\sigma, d)$ be its expected reward.*

Let $\mathcal{L}^n$ be the set of ordered partitions returned by Algorithm 4 when run with parameters $\epsilon, MAX, V^*, Y^*, Z^*$ satisfying $MAX \leq OPT$, and $V_1(T^*, Z^*) \geq V^* \geq (1 - \frac{\epsilon}{3})V_1(T^*, Z^*)$. Define $ALG := \max_{(S,Y,T,Z)\in\mathcal{L}^n} V_2'(S, Y, T, Z)$. Then,

$$ALG \geq (1 - \frac{\epsilon}{3})^2 OPT',$$

Proof. Let (S, Y^*, T, Z^*) be an ordered partition. We show that there exists $(S', Y^*, T', Z^*) \in \mathcal{L}^n$ such that $V_2'(S', Y^*, V^*) \geq (1 - \frac{\epsilon}{6n})^n V_2'(S, Y^*, V^*)$ and for which $V_1(T', Z^*) \geq (1 - \frac{\epsilon}{6n})^n V_1(T, Z^*)$.

We prove this by induction. Let (S_j, Y^*, T_j, Z^*) be any ordered partition obtained by restricting S, T to the first j variables $X_1, \ldots, X_j$ considered by the algorithm (here variables are considered by increasing right endpoint so that $b_1 \leq \cdots \leq b_j$). We show that at the end of the iteration j of the algorithm (after the TRIM procedure), there exists $(S_j', Y^*, T_j', Z^*) \in \mathcal{L}^j$ such that

$$\begin{aligned} V_2'(S_j', Y^*, V^*) &\geq \rho^j V_2'(S_j, Y^*, V^*), \\ V_1(T_j', Z^*) &\geq \rho^j V_1(T_j, Z^*), \\ V_1(S_j', Y^*, V^*) &\geq \rho^j V_1(S_j, Y^*, V^*), \end{aligned} \tag{3.5}$$

where $\rho = (1 - \frac{\epsilon}{6n})$.

The base case with $j = 0$ is true automatically. For the induction step, suppose (A.1) is true for j. Given $(S_{j+1}, Y^*, T_{j+1}, Z^*)$, we wish to prove that there exists $(S_{j+1}', Y^*, T_{j+1}', Z^*)$ that satisfies (A.1). By the induction hypothesis, for the partition (S_j, Y^*, T_j, Z^*) where $S_j = S_{j+1}\backslash X_{j+1}$ and $T_j = T_{j+1}\backslash X_{j+1}$ (note that X_{j+1} is in either S_{j+1} or T_{j+1} but not both), there exists a partition $(S_j', Y^*, T_j', Z^*) \in \mathcal{L}^j$ that satisfies (A.1). In the beginning of iteration $j + 1$, the algorithm will add

two partitions $(\{X_{j+1}, S'_j\}, Y^*, T'_j, Z^*)$ and $(S'_j, Y^*, \{X_{j+1}, T'_j\}, Z^*)$ to $\mathcal{L}^{j+1}$. We claim that one of these two partitions will satisfy the required induction statement for j+1 but with a better factor ρ^j instead of the required ρ^{j+1}.

Depending on whether X_{j+1} appears in T_{j+1} or S_{j+1}, we can consider two cases: either $T_{j+1} = T_j$ or $S_{j+1} = S_j$. In the first case (when $T_{j+1} = T_j$ so $X_{j+1} \in S_{j+1}$), we set $(S'_{j+1}, Y^*, T'_{j+1}, Z^*)$ as the partition $(\{X_{j+1}, S'_j\}, Y^*, T'_j, Z^*)$. By the induction hypothesis $V_1(T'_{j+1}, Z^*) = V_1(T'_j, Z^*) \geq \rho^j V_1(T_j, Z^*) = \rho^j V_1(T_{j+1}, Z^*)$ and

$$
\begin{aligned}
V_1(S'_{j+1}, Y^*, V^*) &= V_1(\{X_{j+1}, S'_j\}, Y^*, V^*) && (3.6)\\
&= V_1(X_{j+1}, V_1(S'_j, Y^*, V^*)) && (3.7)\\
&\geq V_1(X_{j+1}, \rho^j V_1(S_j, Y^*, V^*)) && (3.8)\\
&\geq \rho^j V_1(X_{j+1}, S_j, Y^*, V^*) && (3.9)\\
&= \rho^j V_1(S_{j+1}, Y^*, V^*) && (3.10)
\end{aligned}
$$

where (3.8) follows from the induction hypothesis.

$$
\begin{aligned}
V'_2(S'_{j+1}, Y^*, V^*) &= V'_2(\{X_{j+1}, S'_j\}, Y^*, V^*) && (3.11)\\
&= p_{j+1}(b_{j+1} + V_1(S'_j, Y^*, V^*)) + (1 - p_{j+1})V'_2(S'_j, Y^*, V^*) && (3.12)\\
&\geq p_{j+1}(b_{j+1} + \rho^j V_1(S_j, Y^*, V^*)) + (1 - p_{j+1})\rho^j V'_2(S_j, Y^*, V^*) && (3.13)\\
&\geq \rho^j [p_{j+1}(b_{j+1} + V_1(S_j, Y^*, V^*)) + (1 - p_{j+1})V'_2(S_j, Y^*, V^*)] && (3.14)\\
&= \rho^j V'_2(X_{j+1}, S_j, Y^*, V^*) && (3.15)\\
&= \rho^j V'_2(S_{j+1}, Y^*, V^*) && (3.16)
\end{aligned}
$$

where (3.12) and (3.15) are from the fact that we assume the policy is in P', and (3.13) is from the induction hypothesis.

For the second case (when $S_{j+1} = S_j$ so $X_{j+1} \in T_{j+1}$), we set $(S'_{j+1}, Y^*, T'_{j+1}, Z^*) = (S'_j, Y^*, \{X_{j+1}, T'_j\}, Z^*)$, so that by induction hypothesis, $V_1(S'_{j+1}, Y^*, V^*) = V_1(S'_j, Y^*, V^*) \geq \rho^j V_1(S_j, Y^*, V^*) = \rho^j V_1(S_{j+1}, Y^*, V^*)$, and $V'_2(S'_{j+1}, Y^*, V^*) = V'_2(S'_j, Y^*, V^*) \geq \rho^j V'_2(S_j, Y^*, V^*) = \rho^j V'_2(S_{j+1}, Y^*, V^*)$. Also,

$$\begin{aligned} V_1(T'_{j+1}, Z^*) = V_1(X_{j+1}, T'_j, Z^*) &= V_1(X_{j+1}, V_1(T'_j, Z^*)) && (3.17)\\ &\geq V_1(X_{j+1}, \rho^j V_1(T_j, Z^*)) && (3.18)\\ &\geq \rho^j V_1(X_{j+1}, T_j, Z^*) && (3.19)\\ &= \rho^j V_1(T_{j+1}, Z^*) && (3.20) \end{aligned}$$

where (3.18) is from the induction hypothesis.

However, one or both of these two partitions may be removed by the TRIM procedure. We claim that if any of the two partitions $(S'_{j+1}, Y^*, T'_{j+1}, Z^*) \in \{(\{X_{j+1}, S'_j\}, Y^*, T'_j, Z^*), (S'_j, Y^*, \{X_{j+1}, T'_j\}, Z^*)\}$ is removed by the TRIM procedure, then there will remain another partition $(S''_{j+1}, Y^*, T''_{j+1}, Z^*)$ in $\mathcal{L}^{j+1}$ satisfying:

$$\begin{aligned} V'_2(S''_{j+1}, Y^*, V^*) &\geq V'_2(S'_{j+1}, Y^*, V^*),\\ V_1(T''_{j+1}, Z^*) &\geq \rho V_1(T'_{j+1}, Z^*),\\ V_1(S''_{j+1}, Y^*, V^*) &\geq \rho V_1(S'_{j+1}, Y^*, V^*), \end{aligned} \tag{3.21}$$

To see (A.4), note that $(S'_{j+1}, Y^*, T'_{j+1}, Z^*)$ falls in some bucket $\mathcal{B}_{j_1,j_2}$ during the TRIM procedure. Thus, the TRIM procedure will select one partition from this bucket, let it be (S''_{j+1}, T''_{j+1}). By the criteria for selecting a partition from a bucket, we have $V'_2(S''_{j+1}, Y^*, V^*) \geq V'_2(S'_{j+1}, Y^*, V^*)$, and by construction of buckets, we have both $V_1(T''_{j+1}, Z^*) \geq \rho V_1(T'_{j+1}, Z^*)$ and $V_1(S''_{j+1}, Y^*, V^*) \geq \rho V_1(S'_{j+1}, Y^*, V^*)$.

This proves the induction statement. Applying to an optimal partition (S^*, Y^*, T^*, Z^*), we get that there exists $(S', Y^*, T', Z^*) \in \mathcal{L}^n$ s.t. $V'_2(S', Y^*, V^*) \geq \rho^n V'_2(S^*, Y^*, V^*)$ and $V_1(T', Z^*) \geq \rho^n V_1(T^*, Z^*)$. Coupled with the fact that $V_1(T^*, Z^*) \geq V^*$, we have that $V_1(T', Z^*) \geq \rho^n V^*$.

Putting this together:

$$
\begin{aligned}
V_2'(S', Y^*, T', Z^*) &= V_2'(S', Y^*, V_1(T', Z^*)) && (3.22)\\
&\geq \rho^n V_2'(S', Y^*, V^*) && (3.23)\\
&\geq \rho^{2n} V_2'(S^*, Y^*, V^*) && (3.24)\\
&\geq (1 - \frac{\epsilon}{6n})^{2n}(1 - \frac{\epsilon}{3}) V_2'(S^*, Y^*, T^*, Z^*) && (3.25)\\
&\geq (1 - \frac{\epsilon}{3})^2 V_2'(S^*, Y^*, T^*, Z^*) && (3.26)
\end{aligned}
$$

where (3.22) is due to the fact that after examining Y^*, the player can only accept one more variable. (3.23) is because $V_1(T', Z^*) \geq \rho^n V^*$. (3.24) is from $V_2'(S', Y^*, V^*) \geq \rho^n V_2'(S^*, Y^*, Z^*)$. Finally (3.25) is from our assumption that $V^* \geq (1 - \frac{\epsilon}{3}) V_1(T^*, Z^*)$. □

Lemma 3.4.6. *Let* $OPT_1 := \max_{\sigma \in \mathcal{L}} V_1(\sigma)$. *Algorithm 3 with parameters* $\epsilon \in (0, 1)$ *and* $MAX \geq \frac{OPT_1}{2}$ *runs in time* $O(\frac{n^8}{\epsilon^5})$.

Proof. Given $\text{MAX} \geq \frac{\text{OPT}_1}{2}$, in the TRIM procedure (Algorithm 5), we always have $\frac{max_i}{\text{MAX}} \leq \frac{\text{OPT}_1}{\text{OPT}_1/2} \leq 2$ for $i = 1, 2$. Therefore, the condition $\rho^{J_i} max_i \geq \frac{\epsilon}{3n}\text{MAX}$, $i = 1, 2$ in the TRIM procedure ensures that

$$
J_i \leq \log_{1/\rho}(\frac{3n}{\epsilon}\frac{max_i}{\text{MAX}}) \leq \log_{1/\rho}(\frac{6n}{\epsilon}) = O(\frac{1}{(1-\rho)}\frac{n}{\epsilon}) = O(\frac{n^2}{\epsilon^2})
$$

Therefore, the size of $|\{\mathcal{B}_{j_1,j_2}\}_{j_1,j_2}|$ is $O(\frac{n^4}{\epsilon^4})$. Since for each partition, we need to calculate the expected reward, which is $O(n)$ time, and there are n iterations, Algorithm 4 has a runtime of $O(\frac{n^6}{\epsilon^4})$. Now, Algorithm 3 loops through all pairs of variables, and $J' = O(\frac{1}{\epsilon})$, the total runtime is $O(\frac{n^8}{\epsilon^5})$. □

Proof of Theorem 7 Set parameter $\text{MAX} = \frac{1}{2}E[\max(X_1, \ldots, X_n)]$. Let OPT_1 be defined as in Lemma 3.4.6. Then, we have that $\frac{1}{2}\text{OPT}_1 \leq \text{MAX} \leq \text{OPT}_1$, so by Lemma 3.4.6, Algorithm 3

runs in time $O(\frac{n^8}{\epsilon^5})$. Since in Algorithm 3, we loop through all possible candidates for Y and Z, we will inevitably find Y^*, Z^* in Lemma 3.4.5. In addition, notice that $\frac{\epsilon \text{OPT}}{3n} \leq E[Z^*] \leq V_1(T^*, Z^*) \leq \text{OPT} \leq 2\text{MAX}$. Therefore, if we discretize between $\frac{\epsilon \text{OPT}}{3n}$ and 2MAX, we will eventually find a V^* satisfying $(1 - \frac{\epsilon}{3}) \leq V^* \leq V_1(T^*, Z^*)$. Hence J' is such that $(1 - \frac{\epsilon}{3})^{J'} 2\text{MAX} \leq \frac{\epsilon \text{OPT}}{3n}$ which implies that

$$(1 - \frac{\epsilon}{3})^{J'} \leq \frac{\epsilon}{6n} \frac{\text{OPT}}{\text{MAX}} \leq \frac{\epsilon}{3n}$$

Thus, setting $J' = \lceil \log(\frac{\epsilon}{3n}) / \log(1 - \frac{\epsilon}{3}) \rceil$ is sufficient to find the required V^*. Hence, we will find Y^*, Z^*, V^* in the loop in Algorithm 3 that satisfies Lemma 3.4.5. By Lemmas 3.4.4 and 3.4.5,

$$\text{ALG} \geq (1 - \frac{\epsilon}{3})^2 \text{OPT}' \geq (1 - \frac{\epsilon}{3})^3 \text{OPT} \geq (1 - \epsilon)\text{OPT}$$

3.5 Prophet Inequalities

In this section, we prove two prophet inequalities for two-point distributions, one for static policies and one for dynamic policies.

3.5.1 Static Prophet Inequality

We provide an algorithm that computes a static ordering that gives a $1 - O(\frac{(\log(k))^{1/4}}{k^{3/4}})$ approximation to the offline maximum, where the offline maximum refers to the sum of the largest k realized values. We denote the offline optimal set to be the set of k random variables with the largest realized values.

We will repeatedly use the following lemma, which is a direct application of Chernoff bounds:

Lemma 3.5.1. *Let $Y_1, \ldots, Y_n$ be independent Bernoulli random variables and let $S := \sum_{i=1}^n Y_i$. Let $\mu := E[S]$ and suppose $\mu \leq k - 2\sqrt{k \log(k)}$. Then $P(\mu - 2\sqrt{k \log(k)} \leq S \leq \mu + 2\sqrt{k \log(k)}) \geq 1 - \frac{2}{k^2}$.*

Proof. This is a straightforward application of Chernoff bounds:

$$\begin{aligned} P(|S-\mu| \geq 2\sqrt{k\log(k)}) &\leq 2\exp(-\frac{4k\log(k)}{2\mu+2\sqrt{k\log(k)}}) \\ &= 2\exp(-\frac{2k\log(k)}{\mu+\sqrt{k\log(k)}}) \\ &\leq 2\exp(-\frac{2k\log(k)}{k-\sqrt{k\log(k)}}) \\ &\leq \frac{2}{k^2} \end{aligned}$$

□

Algorithm 6 Static Prophet Inequality

Input: Random two-point variables $X_1, \ldots, X_n$, parameter k for number of rewards player can accept.
Define $\delta = 4\sqrt{k\log(k)}$,
$t_1 := \max\{t : k - 2\sqrt{k\log(k)} - 1 \leq E[\sum_{i=1}^{n} \mathbb{1}(X_i \geq t)]\}$,
R is set of random variables with the k largest left endpoints, $S = \{X_i\}_{i=1}^{n} \backslash R$
$P_1 := \{X_i : X_i \in S, b_i \geq t_1\}$, $P_2 := \{X_i : X_i \in R, b_i \geq t_1\}$, $P_3 := \{X_i\}_{i=1}^{n} \backslash (P_1 \cup P_2)$

Phase 1
Examine the variables in P_1 in any order. Accept X_i iff $X_i \geq t_1$.
Examine the variables in P_2 in increasing a_i. Wlog, assume the order is $X_1, \ldots, X_p$.
for all $i = 1, \ldots, p$ **do**
  if $X_i \geq t_1$ **then**
    Accept X_i.
  end if
  if $X_i < t_1$ **then**
    Define k_1 is the number of remaining rewards the player can accept, $k_2 := p - i + 1$ (p is the total number of variables in P_2).
    Accept X_i iff $k_1 - k_2 \geq E[\sum_{X_i \in P_3} \mathbb{1}(X_i \geq a_i)] + 2\sqrt{\delta\log(\delta)}$.
  end if
end for

Phase 2
Define $t_2 := \max\{t : j - 2\sqrt{\delta\log(\delta)} - 1 \leq E[\sum_{X_i \in P_3} \mathbb{1}(X_i \geq t)]\}$, where j is the number of slots remaining.
Examine the variables in P_3 in any order. Accept X_i iff $X_i \geq t_2$.

We provide an outline for an algorithm which achieves the desired prophet inequality. The full details are in Algorithm 6. First, we select a threshold t_1 such that if a variable realizes a reward that is at least t_1, the variable will be in the offline optimal set with "high probability." Define $Z_1(t) = \sum_{i=1}^{n} \mathbb{1}(X_i \geq t)$, which is the number of random variables with a realized value greater than t. We choose the largest t_1 such that $k - 2\sqrt{k \log(k)} - 1 \leq E[Z_1(t_1)]$. Using our assumption that support points are distinct, this implies that $E[Z_1(t_1)] \leq k - 2\sqrt{k \log(k)}$ (we will provide details in the proof). Lemma 3.5.1 then implies that the probability that $Z_1 \leq k$ is high (at least $1 - \frac{2}{k^2}$), hence any variable that realizes a value of at least t_1 is in the offline optimal set with high probability.

Let P_1, P_2, P_3, R be defined as in Algorithm 6. R is a set containing variables with a large LEP, P_1 is a set with small LEP and large REP variables, P_2 is a set with large LEP and large REP variables, and P_3 is all of the remaining. R represents a reserve set and acts as "insurance" because its variables have a high LEP and guarantee a high realized value. Intuitively, P_1 are variables for which we want to accept its REP and reject its LEP. If the number of variables in P_1 that realize its REP is low, then we need to accept more variables in R that realize their LEP, and vice versa. Therefore, R acts as a hedge in the event that many variables in P_1 realize their LEP.

The algorithm first examines all of the variables in P_1 and accepts iff its REP is realized. If the REP is realized, then the variable is in the offline optimal set with high probability, as mentioned earlier. The motivation for rejecting the LEP of a variable in P_1 is because by construction it cannot be in the offline optimal set, since there are at least k variables with a larger LEP (the variables in R).

Next, we examine all of the variables in P_2 in increasing LEP. Again, we accept the variable if its REP is realized, since any REP must be at least t_1. If its LEP is realized, we accept it if we think with high probability that the number of variables in P_3 exceeding the LEP is small (the full details are in Algorithm 6).

Finally, we choose a second threshold t_2 for the remaining variables, such that if a variable realizes a reward that is at least t_1, the variable will be in the offline optimal set with high probabil-

ity. Suppose we accept $k - j$ variables in P_1 and P_2 for some number j, so we can accept j more variables in P_3. By Lemma 3.5.1, with high probability, $j \leq 4\sqrt{k\log(k)} =: \delta$. Define $Z_2(t) := \sum_{X_i \in P_3} \mathbb{1}(X_i \geq t)$. We choose the largest threshold t_2 such that $j - 2\sqrt{\delta\log(\delta)} - 1 \leq E[Z_2(t_2)]$ (we prove that this is possible in the proof below). We examine the variables in P_3 in any order and accept if a variable realizes a value at least t_2.

Theorem 8. *Let ALG denote the reward of the policy defined in Algorithm 6 and let* $\delta := 4\sqrt{k\log(k)}$. *With probability* $1 - O(\frac{1}{k\log(k)})$*, ALG returns at least*

$$\left(\frac{k - 2\sqrt{\delta\log(\delta)}}{k}\right) OPT = \left(1 - O\left(\frac{(\log k)^{1/4}}{k^{3/4}}\right)\right) OPT$$

where OPT is the offline maximum.

Proof. To prove the theorem, it is enough to prove that the policy satisfies the following two conditions with probability $1 - O(\frac{1}{k\log(k)})$:

1. The policy accepts at least $k - 2\sqrt{\delta\log(\delta)}$ variables.

2. All variables larger than the smallest accepted realized variable are also accepted.

Together, these two conditions imply that the policy misses at most the bottom $2\sqrt{\delta\log(\delta)}$ values in the offline maximum. Hence, the policy returns at least

$$\frac{k - 2\sqrt{\delta\log(\delta)}}{k} OPT = \left(1 - O\left(\frac{(\log k)^{1/4}}{k^{3/4}}\right)\right) OPT$$

Now, we will prove that the two conditions hold. There are two events we wish to hold:

1. E_1 which is the event where $k - \delta \leq Z_1(t_1) \leq k$, where t_1 is defined in Algorithm 6.

2. E_2 which is the event where either $j \leq 2\sqrt{\delta\log(\delta)}$ or $j - 4\sqrt{\delta\log(\delta)} \leq Z_2(t_2) \leq j$, where j is the number of slots remaining at the end of Phase 1 and t_2 is defined in Algorithm 6.

Lemma 3.5.2 gives the probability for which events E_1 and E_2 happen. If both events occur, then condition 1 from above is fulfilled.

Lemma 3.5.2. *With probability* $1 - O(\frac{1}{k \log(k)})$*,* E_1 *and* E_2 *both occur, which implies that the policy accepts between* $k - 2\sqrt{\delta \log(\delta)}$ *and* k *variables.*

Proof. Since we have chosen t_1 such that $k - 2\sqrt{k \log(k)} - 1 \leq E[Z_1(t_1)] \leq k - 2\sqrt{k \log(k)}$, by Lemma 3.5.1, $P(E_1) = P(k - \delta \leq Z_1(t_1) \leq k) \geq 1 - \frac{2}{k^2}$.

Now, we calculate $P(E_2|E_1)$. Since E_1 holds, $0 \leq j \leq \delta$. If it is possible to set t_2 such that $j - 2\sqrt{\delta \log(\delta)} - 1 \leq E[Z_2] \leq j - 2\sqrt{\delta \log(\delta)}$, then we can apply Lemma 3.5.1 again to get $P(E_2|E_1) = P(j - 4\sqrt{\delta \log(\delta)} \leq Z_2(t_2) \leq j) \geq 1 - \frac{2}{\delta^2} = 1 - \frac{1}{8k \log(k)}$. Hence,

$$\begin{aligned} P(E_1 \cap E_2) &= P(E_1)P(E_2|E_1) \\ &= \left(1 - \frac{2}{k^2}\right)\left(1 - \frac{1}{8k \log(k)}\right) \\ &= 1 - O\left(\frac{1}{k \log(k)}\right) \end{aligned}$$

It remains to show that there exists such a threshold t_2 such that $j - \sqrt{\delta \log(\delta)} - 1 \leq E[Z_2(t_2)] \leq j - \sqrt{\delta \log(\delta)}$. We consider two cases:

1. If the policy does not reject a variable in P_2, then there must be at least j variables in R present in P_3. Suppose we gather all of the support points in P_3, which we define as $S := \bigcup_{X_i \in P_3} \{a_i, b_i\}$, where a_i, b_i are the LEP and REP of X_i. W.l.o.g. assume $S := \{s_1, \ldots, s_q\}$, and they are ordered in increasing sequence by index ($s_i < s_{i'}$ for $i < i'$) and further define $s_0 = 0$. We know that $E[Z_2(s_0)] \geq j$ since there are at least j variables in P_3, and also $E[Z_2(s_q)] \leq 1$. In addition, $E[Z_2(s_i)] = E[Z_2(s_{i+1})] + P(X_{i'} = s_i)$, where $X_{i'}$ is the random variable whose support contains s_i, and this uses our assumption that support points are distinct. Therefore, we have $E[Z_2(s_i)] \geq E[Z_2(s_{i+1})] \geq E[Z_2(s_i)] - 1$. Hence, there must exist some support point such that setting t_2 at the support point gives us our required threshold.

2. If the policy does reject a variable in P_2, let a_i be the largest LEP that is rejected. Then this

implies that right before rejecting X_i, $k_1 - k_2 \leq E[Z_2(a_i)] + 2\sqrt{\delta \log(\delta)}$, where k_1 and k_2 are defined in Algorithm 6 before X_i is rejected. Now, since a_i is the last LEP that is rejected in P_2, all remaining variables in P_2 are accepted. Thus, $k_2 - 1$ additional variables are accepted, so the total number of slots remaining after examining P_2 is $j = k_1 - k_2 + 1$. Then we have $j - 2\sqrt{\delta \log(\delta)} - 1 = k_1 - k_2 - 2\sqrt{\delta \log(\delta)} \leq E[Z_2(a_i)]$. Again, to find a threshold t_2 such that $E[Z_2(t_2)] \leq j - 2\sqrt{\delta \log(\delta)}$, we can reason similarly as in part 1 to show that setting t_2 at some support point in $s_1, \ldots, s_q$ works.

□

The following lemma will help prove the second condition.

Lemma 3.5.3. *Suppose $X_1, \ldots, X_p$ are the variables in P_2 and they are ordered by increasing LEP. If some variable $X_{i'}$ realizes its LEP and the policy accepts $X_{i'}$, then the policy accepts $X_{i'+1}, \ldots, X_p$.*

Proof. For notation, let $k_1^{(i)}$ and $k_2^{(i)}$ be the values for k_1, k_2 in step $2b$ when the variable X_i is examined. Then, if $X_{i'} = a_{i'}$ is accepted, it must be that $k_1^{(i')} - k_2^{(i')} \geq E[Z_2(a_{i'})] + 2\sqrt{\delta \log(\delta)}$. Now, because $a_j \geq a_{i'}$ for all $j > i'$, we have $E[Z_2(a_j)] + 2\sqrt{\delta \log(\delta)} \leq E[Z_2(a_{i'})] + 2\sqrt{\delta \log(\delta)}$ since $E[Z_2(a_j)] \leq E[Z_2(a_{i'})]$. It suffices to prove that $k_1^{(j)} - k_2^{(j)} \geq k_1^{(i')} - k_2^{(i')}$ for all $j \geq i'$. This follows from the observation that k_2 decreases by one for each variable in P_2 we examine. However, k_1 only decreases if a variable is accepted. Thus, the difference $k_1 - k_2$ can only increase. Putting it together, we have $k_1^{(j)} - k_2^{(j)} \geq E[Z_2(a_j)] + 2\sqrt{\delta \log(\delta)}$ for all $j \geq i'$. This proves that if the policy examines X_j for $j > i'$, then it accepts X_j. □

Now we will prove the second condition.

Lemma 3.5.4. *If E_1 and E_2 hold, then all variables larger than the smallest accepted realized variable are also accepted.*

Proof. First, if the smallest realized variable accepted is at least t_1, then since E_1 holds, all of the variables greater than t_1 are also accepted and the lemma holds.

Suppose the smallest accepted variable is a LEP a_i in P_2. Then, because we accepted a_i, we have that $k_1 - k_2 \geq E[Z_2(a_i)] + 2\sqrt{\delta \log(\delta)}$, where k_1, k_2 are the values right before examining X_i. Since E_1 holds, we examine all variables in P_1 and P_2, and by Lemma 3.5.3, all random variables in P_2 examined after X_i are accepted, so all larger values in P_1 and P_2 are accepted. Because of this, $j = k_1 - k_2 \geq E[Z_2(a_i)] + 2\sqrt{\delta \log(\delta)}$, where j is the number of slots remaining at the beginning of Phase 2.

If the smallest realized variable accepted is a LEP a_i in P_2, then by Lemma 3.5.3, all random variables with a larger LEP than a_i in P_2 are also accepted. Therefore, all larger realized values in P_1 and P_2 are accepted. Since we accepted a_i, we have that $k_1 - k_2 \geq E[Z_2(a_i)] + 2\sqrt{\delta \log(\delta)}$, where k_1, k_2 are the values right before examining X_i. By Lemma 3.5.3, all random variables in P_2 examined after X_i are accepted, hence $j = k_1 - k_2$, and $E[Z_2(a_i)] \leq j - 2\sqrt{\delta \log(\delta)}$. Then by the definition of t_2, we have that $t_2 \leq a_i$ and since E_2 holds, every realized variable in P_3 that is at least t_2 is accepted. This implies that all realized variables larger than a_i are accepted.

Finally, suppose the smallest realized accepted variable is in P_3. Since E_2 holds, all larger variables in P_3 are accepted. We just need to show that the largest LEP we reject in P_2, which we denote as a_i, is smaller than this smallest realized variable in P_3. Because a_i is the largest LEP we reject in P_2, all remaining variables in P_2 are accepted. Let k_1, k_2 be the values right before rejecting X_i. Then, $j - 1 = k_1 - k_2 \leq E[Z_2(a_i)] + 2\sqrt{\delta \log(\delta)} \Rightarrow j - 2\sqrt{\delta \log(\delta)} - 1 \leq E[Z_2(a_i)]$. Since t_2 is chosen to be the largest threshold such that $j - 2\sqrt{\delta \log(\delta)} - 1 \leq E[Z_2(t_2)] \leq j - 2\sqrt{\delta \log(\delta)}$, it must be the case that $t_2 \geq a_i$. Hence, any variable we accept in P_3 is larger than a_i. □

Combining Lemmas 3.5.2 and 3.5.4 gives us the result. □

3.5.2 Dynamic Prophet Inequality

In this section, we provide an algorithm that computes a dynamic policy and returns the $k - 1$ largest values in the optimal offline set. This directly implies a $1 - 1/k$ approximation to the optimal offline. Additionally, we show that this approximation is tight in the limit as $k \to \infty$.

Let $T = \{X_1, \ldots, X_n\}$ be the entire set of random variables, let R be the set of random variables with the k largest LEP, and let $S = T \backslash R$. Define $b_{\min} := \min_{X_i \in R} b_i$, $a_{\min} := \min_{X_i \in R} a_i$, $a_{\max} := \max_{X_i \in R} a_i$, $S^1 := \{X_i : b_i > b_{\min}, X_i \in S\}$, and $S^2 := \{X_i : b_{\min} > b_i > a_{\min}, X_i \in S\}$. We never accept the variables that are not in R, S^1, or S^2 since those variables cannot appear in the optimal offline set, therefore we effectively ignore them.

We give an outline of Algorithm 7, which attains the desired result. For any realization, the algorithm returns the largest $k - 1$ realized variables. If a variable cannot be in the optimal offline, the algorithm rejects it. One consequence of this is that if a variable in S realizes its LEP, it cannot be in the optimal offline, since there are at least k variables with a larger LEP (the variables in R). Therefore, a variable in S is rejected if its LEP is realized, and effectively only its REP matters for the algorithm.

The algorithm is run recursively, so w.l.o.g. suppose that T is the set of variables remaining, and we have k slots remaining. Suppose first that $S^1 \neq \emptyset$. Consider the variable in S^1 with the largest REP and let us call it X. X has the largest REP among all of the variables in S, and furthermore there are at most $k - 1$ variables in R that have a larger REP. If X realizes its REP, then it must be in the optimal offline, and if it realizes its LEP, then it cannot be in the optimal offline (since there are k variables in R that are guaranteed to realize a higher value). Hence, if $S^1 \neq \emptyset$, the algorithm has an obvious choice, which is to examine the variable with the largest REP and accept iff its REP is realized.

If $S^1 = \emptyset$, the variables with the largest REP are all in R. However, now there is a tradeoff between examining a variable in R first versus examining it later, because its LEP may potentially be in the optimal offline. If a variable in R is examined next and its REP is realized, it is in the optimal offline, but if its LEP is realized, it may or may not be in the optimal offline. It may be better, in hindsight, to reserve this variable for later. To manage this tradeoff, the algorithm probes the variable in R with the smallest left endpoint; let us call this X. If the LEP of X is realized, we reject it, and then start probing the variables in S^2 in decreasing REP, accepting iff REP is realized. If a REP in S^2 is realized, the algorithm accepts, and at this point, we know that X could not have

been in the optimal offline, so it was correct in hindsight to have rejected X. On the other hand, if no variables in S^2 realize their REP, then in hindsight it was wrong to have rejected X. However, at this point, only R is non-empty and all variables in R are in the optimal offline and none of the variables in S^2 that the algorithm has rejected could have been in the optimal offline. We can prove that X is the only variable in the optimal offline that the algorithm fails to accept, and in fact, is the smallest variable in the optimal offline.

Throughout this section, define $OPT(T, k)$ to denote the optimal offline set for set T, i.e. the k largest realized values in set T.

The next lemma formalizes several cases when a variable is in the optimal offline. It states that:

1. If a variable is accepted in an iteration, then it must be in the optimal offline set.
2. If a variable is rejected in an iteration, then it is not in the optimal offline set if it is in S.
3. If a variable is rejected and it is in R, then it is not in the optimal offline set iff a later variable is accepted in that iteration (this occurs if line 15 is triggered and a variable is accepted in line 20 in the while loop in Algorithm 7).

Lemma 3.5.5. *Let T_j, Y_j, S_j, R_j, k_j be defined as in Algorithm 7. Suppose we run $DPI(T_1, k_1)$ for some T_1 and k_1. Then we have the following properties:*

1. *If $Y_j \neq \emptyset$, then $Y_j \in OPT(T_j, k_j)$*
2. *If $Z \in T_j \backslash (T_{j+1} \cup Y_j)$ and $Z \in S_j$, then $Z \notin OPT(T_j, k_j)$*
3. *If $Z \in T_j \backslash (T_{j+1} \cup Y_j)$ and $Z \in R_j$, then $Z \notin OPT(T_j, k_j)$ iff $Y_j \neq \emptyset$*

Proof. We prove the first statement, which states that if a variable is accepted, it must be in the optimal offline set. Suppose $Y_j \neq \emptyset$. Y_j can only be accepted in lines 8, 10, 14, 20, or 28. We will show that in any of these cases, $Y_j \in \text{OPT}(T_j, k_j)$.

1. Line 8: In this case Y_j is larger than all of the variables in S_j. Also, $Y_j \in R_j$ and since there are only an additional $k_j - 1$ variables in R_j, this means that $Y_j \in \text{OPT}(T_j, k_j)$.

2. Line 10: Since Y_j is the largest REP in S_j, Y_j is larger than all of the remaining variables in S_j. Since $Y_j \in S_j^1$, Y_j must be larger than at least one variable in R_j (the variable with REP $b_{\min}$). Hence, $Y_j \in \text{OPT}(T_j, k_j)$.

3. Line 14: Since $S_j^1 = \emptyset$, the variables in R_j have the k_j largest REP. Therefore, $Y_j \in \text{OPT}(T_j, k_j)$.

4. Line 20: Y_j must be larger than all of the variables in S_j. Furthermore, Y_j is larger than the variable in R_j with LEP $a_{\min}$ since line 15 must have occurred and $Y \in S_j^2$, meaning its REP is larger than $a_{\min}$. Therefore, $Y_j \in \text{OPT}(T_j, k_j)$.

5. Line 28: If $S_j^1 \cup S_j^2 = \emptyset$, then $\text{OPT}(T_j, k_j) = R_j$. Hence, $Y_j \in \text{OPT}(T_j, k_j)$.

We now prove the second statement. $T_j \backslash (T_{j+1} \cup Y_j)$ are the variables that are rejected before line 31. Hence, if $Z \in S_j$, then Z must realize its LEP. Now, all of the variables in R_j are guaranteed to be larger than Z and $|R_j| = k_j$, so $Z \notin \text{OPT}(T_j, k_j)$.

Finally, we prove the third statement. If $Z \in R_j$, then Z must be the variable rejected in line 16. This implies that Z has the smallest LEP in R_j and its LEP is realized. If $Y_j \neq \emptyset$, then $Y_j \in S_j^2$, so some variable in S_j^2 realizes its REP. This means that $Z \notin \text{OPT}(T_j, k_j)$ since Z is smaller than Y_j and the remaining $k-1$ variables in R_j. If $Y_j = \emptyset$, then this means that all variables in S_j^2 realize their LEP and are rejected. None of those variables can be in $\text{OPT}(T_j, k_j)$ by part 2. Hence, the k_j variables in R_j are in $\text{OPT}(T_j, k_j)$, implying that $Z \in \text{OPT}(T_j, k_j)$. □

Theorem 9. *$OPT(T, k-1) \subset DPI(T, k)$, i.e. Algorithm 7 returns at least the largest $k-1$ variables in T.*

Proof. Let $T_1 := T$ and $k_1 := k$. Let A be the event where the while loop in line 17 is triggered, but line 19 is not triggered, i.e. all variables in S_j^2 are examined and rejected. We will prove the theorem by considering the case when A occurs sometime when running $DPI(T, k)$ and when A does not occur.

First suppose that A does not occur. The algorithm produces a sequence of sets $T_1, T_2, \ldots, T_p$. We will prove the (stronger) statement that $\text{OPT}(T_j, k_j) = \text{DPI}(T_j, k_j)$ for all $j \leq p$ using induction on j. The base case is to prove $\text{OPT}(T_p, k_p) = \text{DPI}(T_p, k_p)$, but this is trivial since it must be the case that either $k_p = 0$ or $R_p = \emptyset$.

For the induction step, we want to prove that $\text{OPT}(T_i, k_i) = \text{DPI}(T_i, k_i)$ and assume that $\text{OPT}(T_{i'}, k_{i'}) = \text{DPI}(T_{i'}, k_{i'})$ for all $i' > i$. From the algorithm, $\text{DPI}(T_i, k_i) = \text{DPI}(T_{i+1}, k_{i+1}) \cup Y_i$. By the induction hypothesis, $\text{OPT}(T_{i+1}, k_{i+1}) \cup Y_i = \text{DPI}(T_{i+1}, k_{i+1}) \cup Y_i = \text{DPI}(T_i, k_i)$. To finish the proof, we need to prove $\text{OPT}(T_i, k_i) = \text{OPT}(T_{i+1}, k_{i+1}) \cup Y_i$. We consider two cases, when $Y_i \neq \emptyset$ and when $Y_i = \emptyset$.

1. Suppose $Y_i \neq \emptyset$, which implies that $k_{i+1} = k_i - 1$. From Lemma 3.5.5 parts 2 and 3, $(T_i \backslash (T_{i+1} \cup Y_i)) \cap \text{OPT}(T_i, k_i) = \emptyset$. This implies that $\text{OPT}(T_i, k_i) = \text{OPT}(T_{i+1} \cup Y_i, k_i)$. Furthermore, Lemma 3.5.5 part 1 implies that $Y_i \in \text{OPT}(T_i, k_i)$, thus $\text{OPT}(T_{i+1} \cup Y_i, k_i) = \text{OPT}(T_{i+1}, k_{i+1}) \cup Y_i$. Hence, $\text{OPT}(T_i, k_i) = \text{OPT}(T_{i+1}, k_{i+1}) \cup Y_i$

2. Suppose $Y_i = \emptyset$, which implies $k_{i+1} = k_i$. We need to prove that $\text{OPT}(T_i, k_i) = \text{OPT}(T_{i+1}, k_{i+1})$, i.e. every variable in $T_i \backslash T_{i+1}$ is not in $\text{OPT}(T_i, k_i)$. This is a result of part 2 of Lemma 3.5.5 and also from our assumption that A does not occur (which implies we cannot have both $Y_i = \emptyset$ and $Z \in R_i$ in part 3 of Lemma 3.5.5).

Now we will assume that A occurs. A can only occur once since afterwards $S_i^1 \cup S_i^2 = \emptyset$. W.l.o.g. assume that this occurs when $\text{DPI}(T_m, k_m)$ is run for some m. Some variable is rejected during this run in line 16, which we call Y'. We will prove that $\text{OPT}(T_j, k_j - 1) \subset \text{DPI}(T_j, k_j)$ and also Y' is the smallest variable in $\text{OPT}(T_j, k_j)$ for all $j \leq m$ using induction on j.

The base case is to prove that $\text{OPT}(T_m, k_m - 1) \subset \text{DPI}(T_m, k_m)$. Since A occurs, notice that $Y_m = \emptyset$ and every element in S_m^2 is rejected. Therefore in the next iteration, we have $S_{m+1}^1 \cup S_{m+1}^2 = \emptyset$, $R_m \subset R_{m+1}$ (there will be potentially one additional variable in R_{m+1}), and $k_{m+1} = k_m$. Thus, $\text{DPI}(T_{m+1}, k_{m+1})$ goes straight to line 28 and the algorithm concludes by accepting every element in R_{m+1}, so $\text{DPI}(T_m, k_m) = R_{m+1}$. Furthermore, $Y' < Z$ for all $Z \in R_m$ because Y' has the smallest

LEP in R_m and realized its LEP. By Lemma 3.5.5, $(T_m \backslash T_{m+1}) \cap \text{OPT}(T_m, k_m) = \emptyset$. Therefore, the largest $k_m - 1$ elements in T_m are the variables in $R_m \backslash Y'$, so $\text{OPT}(T_m, k_m - 1) = R_m \backslash Y'$. Hence, $\text{OPT}(T_m, k_m - 1) = R_m \backslash Y' \subset R_{m+1} = \text{DPI}(T_m, k_m)$. By Lemma 3.5.5, $Y' \in \text{OPT}(T_m, k_m)$ and since we proved that Y' is not in $\text{OPT}(T_m, k_m - 1)$, it must be the smallest variable in $\text{OPT}(T_m, k_m)$.

For the induction step, we want to prove that $\text{OPT}(T_i, k_i - 1) \subset \text{DPI}(T_i, k_i)$ and Y' is the smallest element in $\text{OPT}(T_i, k_i)$. By the induction hypothesis, $\text{DPI}(T_i, k_i) = \text{DPI}(T_{i+1}, k_{i+1}) \cup Y_i \supset \text{OPT}(T_{i+1}, k_{i+1} - 1) \cup Y_i$. Hence, it's sufficient if we can prove that $\text{OPT}(T_i, k_i - 1) = \text{OPT}(T_{i+1}, k_{i+1} - 1) \cup Y_i$. We consider two cases, when $Y_i \neq \emptyset$ and when $Y_i = \emptyset$.

1. Suppose $Y_i \neq \emptyset$, which implies that $k_{i+1} = k_i - 1$. By the induction hypothesis, Y' is the smallest variable in $\text{OPT}(T_{i+1}, k_{i+1}) = \text{OPT}(T_{i+1}, k_i - 1)$. By Lemma 3.5.5, $(T_i \backslash (T_{i+1} \cup Y_i)) \cap \text{OPT}(T_i, k_i) = \emptyset$. Therefore, Y' is the smallest variable in $\text{OPT}(T_i \backslash Y_i, k_i - 1)$. Now, we will prove that $Y_i > Y'$, which will prove that Y' is the smallest variable in $\text{OPT}(T_i, k_i)$. Y_i can be accepted in lines 8, 10, 12, 14, or 20.

   (a) Line 8: If Y_i is accepted here, notice that the LEP of Y_i must be larger than the LEP of Y'. If not, then Y' would be examined here instead of Y_i, a contradiction. Hence, $Y_i > Y'$.

   (b) Line 10: If Y_i is accepted here, then this implies that $a_{\max} < Y_i$. Therefore, $Y' < Y_i$.

   (c) Line 14: $a_{\max} < \max\{b_i : X_i \in S_j^2\}$ because we are not at line 7. Since $S_j^1 = \emptyset$, $Y_i \geq b_{\min} > \max\{b_i : X_i \in S_j^2\} > a_{\max} \geq Y'$.

   (d) Line 20: Suppose $Y' > Y_i$. Then this implies that the LEP of $Y' > \max\{b_i : X_i \in S_p^1 \cup S_p^2\}$, hence Y' should be examined in line 8 instead of line 16, a contradiction.

   Now since we know that Y' is the smallest variable in $\text{OPT}(T_i, k_i)$, it follows that $Y_i \in \text{OPT}(T_i, k_i - 1)$. Furthermore, $\text{OPT}(T_{i+1}, k_{i+1} - 1) = \text{OPT}(T_i \backslash Y_i, k_i - 2)$. Therefore, $\text{OPT}(T_{i+1}, k_{i+1} - 1) \cup Y_i = \text{OPT}(T_i, k_i - 1)$.

2. Suppose $Y_i = \emptyset$, which implies $k_{i+1} = k_i$. We need to prove that $\text{OPT}(T_i, k_i - 1) =$

$\text{OPT}(T_{i+1}, k_i - 1)$. This follows from Lemma 3.5.5 which tells us that $(T_i \backslash T_{i+1}) \cap OPT(T_i, k_i) = \emptyset$.

Combining all of this together, we can conclude that $\text{OPT}(T_1, k_1 - 1) \subset \text{DPI}(T_1, k_1)$. □

Corollary 3.5.6.

$$DPI(T, k) \geq (1 - \frac{1}{k}) OPT(T, k)$$

In addition, $DPI(T, k)$ runs in $O(|T|)$ time.

Proof. This follows straight from Theorem 9 since $\text{OPT}(T, k-1) \subset \text{DPI}(T, k)$. The linear runtime follows since it takes constant time to determine which random variable to examine next. □

Tight Example

In this section we provide an example that shows that Algorithm 7 is asymptotically tight. Let OPT be the value of the optimal offline (i.e. the sum of the k largest variables).

Proposition 3.5.7. *Fix a policy and let its reward be donated as ALG. We have the following upper bound:*

$$\frac{E[ALG]}{E[OPT]} \leq 1 - \frac{1}{6k}$$

Proof. Construct the following $k + 1$ random variables:

1. $X_1, \ldots, X_k$ are IID with support $\{1, 3\}$ w.p. $\{1 - 2^{-1/k}, 2^{-1/k}\}$
2. Y has support $\{0, 2\}$ w.p. $\{1/2, 1/2\}$

We first prove a lemma that helps characterize an optimal policy when applied to this set of random variables. Then we show that the upper bound holds for any optimal policy for this example.

Lemma 3.5.8. *Suppose we have Y and $X_1, X_2, \ldots, X_j$ where $j \leq k$, and there are j slots remaining. Any optimal policy on $\{Y, X_1, X_2, \ldots, X_j\}$ can be transformed into an optimal policy where Y is not examined next.*

Proof. Let ALG_1 be the reward of an optimal policy that examines Y next. Let ALG_2 be the reward of a policy that accepts $X_1, \ldots, X_j$ regardless. We will show that $E[\text{ALG}_2 - \text{ALG}_1] \geq 0$, therefore proving that accepting $X_1, \ldots, X_j$ regardless is also optimal. This policy does not examine Y next, proving the lemma.

Let Z be a random variable denoting the number of variables in $\{X_i\}_{i=1}^{j}$ that realize 3. Note that $\text{ALG}_2 = 3Z + (j - Z) = 2Z + j$.

Let Y be examined next. If $Y = 0$, then it is optimal to reject Y and so $\text{ALG}_1 = \text{ALG}_2$. Now suppose $Y = 2$. If the optimal policy rejects Y, then it returns the same as ALG_2 and we are done. Therefore, suppose the optimal policy accepts Y. After accepting Y, the policy needs to pick $j - 1$ of the variables in $\{X_i\}_{i=1}^{j}$. Here, the optimal policy is to reject the first X_i where $X_i = 1$ and accept the rest of the variables regardless. Hence, $\text{ALG}_1 = 2 + 3\min\{Z, j - 1\} + \max\{j - 1 - Z, 0\}$. If $Z \leq j - 1$, then $\text{ALG}_1 = 1 + j + 2Z$ and $\text{ALG}_2 - \text{ALG}_1 = -1$. If $Z = j$, then $\text{ALG}_1 = 3j - 1$ and $\text{ALG}_2 - \text{ALG}_1 = 1$.

Therefore,

$$
\begin{aligned}
E[\text{ALG}_2 - \text{ALG}_1] &= E[\text{ALG}_2 - \text{ALG}_1 | Y = 0]P(Y = 0) + E[\text{ALG}_2 - \text{ALG}_1 | Y = 2]P(Y = 2) \\
&= \frac{E[\text{ALG}_2 - \text{ALG}_1 | Y = 2]}{2} \\
&= \frac{E[\text{ALG}_2 - \text{ALG}_1 | Y = 2, Z \leq j - 1]P(Z \leq j - 1)}{2} \\
&+ \frac{E[\text{ALG}_2 - \text{ALG}_1 | Y = 2, Z = j]P(Z = j)}{2} \\
&= \frac{P(Z = j) - P(Z \leq j - 1)}{2} \\
&\geq 0
\end{aligned}
$$

The last inequality follows since $P(Z = j) = \prod_{i=1}^{j} 2^{-1/k} = 2^{j/k}$. For $j \leq k$, $2^{j/k} \geq 1/2$, hence

$P(Z = j) \geq P(Z \leq j - 1)$. This proves that $E[\text{ALG}_2] \geq E[\text{ALG}_1]$. □

Lemma 3.5.9. *An optimal policy is to accept all X_i.*

Proof. From Lemma 3.5.8, we limit our consideration to optimal algorithms where X_i is examined next. Suppose we have $X_1, \ldots, X_j$ and Y remaining and we probe X_1 next. If $X_1 = 3$, it is optimal to accept. If $X_1 = 1$, then accepting gives us $(j-1)E[X] + 1$ whereas rejecting gives us $(j-1)E[X] + E[Y] = (j-1)E[X] + 1$. Therefore, accepting or rejecting X_1 gives the same expected reward. By induction, an optimal algorithm is simply to accept all X_i regardless. □

Let ALG denote the reward for accepting all X_i, which is optimal from Lemma 3.5.9, and let OPT be the optimal offline. Let Z denote the number of X_i where $X_i = 3$. Then we have $\text{ALG} = 3Z + (k - Z) = 2Z + k$ and $\text{OPT} = 3Z + 2\min\{k - Z, 1\} + \max\{k - Z - 1, 0\}$. For $Z = k$, $\text{OPT} = 3k$ and $\text{OPT} - \text{ALG} = 0$. For $Z \leq k - 1$, $\text{OPT} = 2Z + k - 1$ and $\text{OPT} - \text{ALG} = 1$. Therefore,

$$\begin{aligned}
\frac{E[\text{OPT} - \text{ALG}]}{E[\text{OPT}]} &= \frac{E[\text{OPT} - \text{ALG} | Z = k]P(Z = k) + E[\text{OPT} - \text{ALG} | Z \leq k - 1]P(Z \leq k - 1)}{E[\text{OPT}]} \\
&\geq \frac{P(Z \leq k - 1)}{3k} \\
&= \frac{1}{6k}
\end{aligned}$$

where we use the fact that $P(Z \leq k - 1) = 1 - P(Z = k) = 1 - (2^{-1/k})^k = 1/2$. Rearranging gives us

$$\frac{E[\text{ALG}]}{E[\text{OPT}]} \leq 1 - \frac{1}{6k}$$

□

Algorithm 7 DPI(T_j, k_j)

Input: Random variables $T_j = \{X_1, \ldots, X_n\}$ and a parameter $k_j \geq 0$
Initialize: R_j is the set of random variables with the k_j largest LEP, $S_j := T_j \backslash R_j$.
$b_{\min} := \min_{X_i \in R_j} b_i$, $a_{\min} := \min_{X_i \in R_j} a_i$, $a_{\max} := \max_{X_i \in R_j} a_i$
$S_j^1 := \{X_i : b_i > b_{\min}, X_i \in S_j\}$ and $S_j^2 := \{X_i : b_{\min} > b_i > a_{\min}, X_i \in S_j\}$
if $k_j = 0$ or $R_j = \emptyset$ **then**
    return $\emptyset$
end if
if $S_j^1 \cup S_j^2 \neq \emptyset$ **then**
    if $a_{\max} > \max\{b_i : X_i \in S_j^1 \cup S_j^2\}$ **then**
        Accept the variable with left endpoint $a_{\max}$.
    else if $S_j^1 \neq \emptyset$ **then**
        Examine the variable in S_j^1 with the largest REP. Accept iff its REP is realized.
    else
        Let Y' be the variable in R_j with left endpoint $a_{\min}$. Examine Y'.
        if REP of Y' is realized **then**
            Accept Y'.
        else if LEP of Y' is realized **then**
            Reject Y'.
            while $S_j^2 \neq \emptyset$ **do**
                Examine $X_i \in S_j^2$ with largest REP.
                if b_i is realized **then**
                    Accept X_i. Break out of while loop.
                else
                    Reject X_i.
                end if
            end while
        end if
    end if
else
    Probe and accept an (arbitrary) variable in R_j
end if
Let T_{j+1} be the set of remaining unexamined variables and let k_{j+1} to be the number of slots remaining. If a variable was accepted, let Y_j be that variable, otherwise set $Y_j = \emptyset$.
return DPI(T_{j+1}, k_{j+1}) $\cup Y_j$

Chapter 4: Popular Matchings

4.1 Introduction

Matching problems lie at the intersection of different areas of mathematics, computer science, and economics, and at the core of each of those. In discrete mathematics, they have been studied at least since Petersen; in discrete optimization, Edmonds' maximum weighted matching algorithm is one the earliest and most remarkable example of non-trivial polynomial-time algorithm; in game theory, network bargaining problems models are tightly connected to maximum matchings and their linear programming formulations; in two-sided markets, Gale-Shapley's model has been widely used and generalized to assign e.g. students to schools and interns to hospitals.

Gale-Shapley's model assumes that the input graph is bipartite, and that each node is endowed with a strict preference list over the set of its neighbors. The goal is to find a matching that respects a certain concept of fairness called *stability*. This model has been generalized in many ways. On one hand, one can change the structure of the *input*, assuming e.g. the presence of ties in the preference lists, or of quotas in one of the two sides of the bipartition, or that the preference pattern is given by more complex choice functions. On the other hand, one can change the requirements of the *output*, i.e. ask that the matching we produce satisfies properties other than stability. For instance, relaxing the stability condition to *popularity* allows to overcome one of the main drawbacks of stable matchings: the fact that two individuals (a *blocking pair*) can prevent the matching from being much larger. Roughly speaking, a matching M is *popular* if, for every other matching M', the number of nodes that prefer M to M' are at least as many as the number of nodes that prefer M' to M (this model is called *two-sided*; formal definitions are given in Section 4.2). One can show that stable matchings are popular matchings of minimum size, and the size of a popular matching can be twice as much as the size of a stable matching (recall that all

stable matchings are maximal, and a maximal matching has size at least half the size of a maximum matching). Hence, popularity allows for matchings of larger size while still guaranteeing a certain fairness condition.

Surprisingly, most of what is known about popular matchings derives in a way or another from stable matchings. For instance, Kavitha [39] showed that a maximum size popular matching can be found efficiently by a combination of repeated applications of the *Gale-Shapley's* algorithm and *promotion*. Cseh and Kavitha [16] showed that a pair of nodes are matched together in at least a popular matching if and only if they are matched together either in some stable matching, or in some *dominant* matching, with the latter being (certain) popular matchings of maximum size, and the image of stable matchings in another graph under a linear map. This fact implies that the *popular edge problem* can be solved in polynomial time (see Section 4.3 for definitions). In [16], it is asked whether their results can be extended to find a popular matching that contains two given edges. Other open questions in the area involve finding a popular matching of maximum (a generalization of [16]) or minimum (and its special case, the *unpopular edge problem*) weights, in particular when weights are nonnegative, see e.g. [16, 15].

Our Contribution. In this chapter, we settle the complexity of all those problems by showing that it is NP-Complete to decide whether there exists a popular matching that contains (or does not contain) two given edges. More generally, we give a very granular analysis of the complexity of the *popular matching with forbidden and forced elements* problem, i.e. deciding whether a popular matching exists when we force it (or forbid it) to have certain edges or vertices.

From our reduction, it follows that the problem of finding a popular matching of *minimum* weight when weights on edges are nonnegative cannot be approximated up to any factor (one can indeed define the objective function so that it is NP-Complete to decide if the optimum is 0). If instead, still under the nonnegativity assumption on the weights, we want to find a popular matching of *maximum* weight, the problem becomes inapproximable to a factor better than $1/2$. It follows from a decomposition theorem for popular matchings given in [16] that a $1/2$ approximation can be achieved in polynomial time by taking the matching of maximum weight from the set given by

dominant and stable. This shows that our result is tight. Our findings suggest that stable matchings and their closely connected counterpart dominant matchings seem to be the only tractable subclasses of popular matchings.

Organization of the chapter. The chapter is organized as follows. In Section 4.2, we formally define the main problems under consideration and state our results. In Section 4.3, we introduce suitable notation and recall results that will be used in proofs. The main technical contribution is given in Section 4.4, where a reduction from monotone 3-SAT to the existence of certain popular matchings is given. In Section 4.5, we combine basic facts showed in Section 4.3 and the reduction from Section 4.4 to investigate the complexity of popular matching with forbidden and forced elements problems. In Section 4.5.3, we use the same reduction idea in Section 4.4 to establish further NP-Hardness results. Finally in Section 4.6 we investigate restricting the graph to have bounded treewidth, which allows a polynomially tractable algorithm for finding the max weight popular matching.

4.2 Problems and main results

Let $G = (V, E)$ be a simple, undirected graph. Throughout this chapter, we denote an edge between nodes u and v as uv. For each vertex $v \in V$, let $\Gamma(v)$ denote the neighbors of v in G and let $<_v$ be a strict total order over $\Gamma(v)$. If $u <_v w$, we say that v *prefers* u to w. We write $u \leq_v w$ if $w = u$ or $u <_v w$, and we say that v *weakly prefers* u to w. The pair $(G, <)$ is called a *preference system*, where $<$ is the collection of strict total orders $<_v$ for $v \in V$.

A *matching* of $(G, <)$ is a set $M \subseteq E$ such that, for each node $v \in V$, $|M \cap \Gamma(v)| \leq 1$. If the previous inequality holds at equality, we say that v is *matched* or *covered* by M. Otherwise, we say that v is *unmatched*, or *exposed by* M, or M*-exposed*. We denote by $V(M)$ the set of nodes matched by M; for $v \in V(M)$, we let $M(v)$ be the node $v \in V$ is matched to in M, while for an M-exposed node $v \in V$ we set $M(v) = \emptyset$.

A vertex $v \in V$ is said to *prefer* matching M to matching M' if either v is matched in M and unmatched in M', or v is matched in both and v prefers its partner in matching M to its partner in

matching M'. For matchings M and M', let $\phi(M, M')$ be the number of vertices that prefer M to M'. If $\phi(M', M) > \phi(M, M')$, then we say M' is *more popular than* M. We say a matching M is *popular* if there is no matching that is more popular than M; in other words, $\phi(M, M') \geq \phi(M', M)$ for all matchings M' in $(G, <)$.

The central problem of this chapter is the following.

Problem 4 (Maximum weight popular matching – `mwp`)**. Given:** *A preference system* $(G, <)$*, weights* $w : E(G) \to \mathbb{R}_+$*.* ***Find****: A popular matching* M *of* $(G, <)$ *that maximizes* $w(M)$*, or conclude that non exists.*

We next introduce three special cases of `mwp`, all restricting the input graph to be bipartite. When no other constraints are added to the problem, we denote it by `mwpb`, where the `b` that has been added to the acronym stands for *bipartite*. If we moreover restrict to $w \geq 0$, we obtain `mppb`, where the `w` of weight has been replaced by the `p` of profits. The third special case is defined below.

Problem 5 (Popular matching with forbidden and forced elements problem – `pmffe`)**. Given:** *A preference system* $(G, <)$*, with* G *bipartite,* $U^{in}, U^{out} \subset V(G)$*,* $F^{in}, F^{out} \subset E(G)$*, such that* $U^{in} \cap U^{out} = \emptyset$*,* $F^{in} \cap F^{out} = \emptyset$*, and if* $uv \in F^{in} \cup F^{out}$ *then* $u, v \notin U^{in} \cup U^{out}$*.* ***Find****: A popular matching of* $(G, <)$ *that contains all vertices of* U^{in}*, all edges of* F^{in}*, no vertices of* U^{out} *and no edges of* F^{out}*, or conclude that such matching does not exist.*

`pmffe` can be easily formulated as a special case of `mwpb`, by giving appropriate weights to the edges.

In Section 4.5, we deal with `mwpb`, `mppb`, and `pmffe`, and show the following.

Theorem 10. *`mppb` and `mwpb` are NP-Hard. Unless P=NP, the following holds:*

- *`mwpb` cannot be approximated in polynomial time up to any factor even if* $w(v) \in \{0, -1\}$ *for all* $v \in V$*;*
- *`mwp` cannot be approximated in polynomial time better than a factor* $\frac{1}{2}$*.*

On the other hand, there is a polynomial-time algorithm that computes a $\frac{1}{2}$*-approximation to `mwp`.*

Theorem 11. *If $U^{out} = F^{in} = F^{out} = \emptyset$ or $U^{in} = F^{in} = F^{out} = \emptyset$, or $|U^{in} \cup U^{out} \cup F^{in} \cup F^{out}| \leq 1$, then* `pmffe` *can be solved in polynomial time. Otherwise, for each other set of fixed values for $|U^{in}|, |U^{out}|, |F^{in}|, |F^{out}|$,* `pmffe` *is NP-Complete.*

Theorem 11 gives a complete characterization of the complexity of `pmffe` as a function of the cardinality of sets U^{in}, U^{out}, F^{in}, F^{out}. It is worth mentioning here that all hardness and hardness of approximation results from Theorem 10 and Theorem 11 are implied from a single reduction, presented in Section 4.4. More consequences of this reduction are also presented in Section 4.5.

We next focus on `mwp` and show the following in Section 4.6.

Theorem 12. *Let $\omega \in \mathbb{N}$ be a fixed constant. Then there is an algorithm that solves* `mwp` *in time $|V|^{O(\omega)}$ when the input graph $G(V, E)$ has treewidth at most ω.*

Let us remark that in the previous result, we are not assuming that the graph is bipartite. A discussion on the techniques and on related problems, as well as definitions on treewidth can be also found in Section 4.6.

Conference version and related work. Results from Section 4.5 have first appeared on Arxiv in [22], and then in the conference paper [23] where also results from Section 4.6 had appeared. With respect to [23], proofs for results from Section 4.5 are obtained via a reduction that is shorter and we believe more direct. Section 4.6 contains all details for the bounded treewidth case, while the corresponding, 3-pages long section in [23] only contained a sketch of the arguments.

We want to stress that the reduction from Section 4.4 gives a general framework to approach complexity questions in this area. In fact, small variations of it can be used to prove many new and known hardness result around stable and popular matchings. Those variations add a constant number of edges and / or nodes to the graph constructed in Section 4.4 (see Figure 4.2). In order to keep the chapter lean, we omit those related reductions and refer the interested reader to the Ph.D. thesis [48] for details. It is worth mentioning that one of those variations can be used to prove the NP-Completeness of the problem of deciding whether there exists a popular matching in a preference system. This hardness result has first been proved in the Arxiv papers [29, 40] and

has then appeared in the conference papers [23, 30]. Further, we remark that other proofs of this and related hardness results have appeared in the subsequent work [17].

4.3 Definitions and basic facts

4.3.1 Popular matchings

Given a preference system $(G, <)$, we often represent $<$ via a *preference pattern*, i.e., a function L that maps ordered pairs (u, v) with $uv \in E$ to the position of v in $<_u$. If $L(u, v) = 1$, we say that v is the *favorite partner* of u.

While a preference pattern completely describes the preference lists, it will sometimes be useful to have only a partial characterization of the preference lists. This is formalized next. An *incomplete preference pattern* I for a bipartite graph G is a map $I : \{(u, v) : uv \in E(G)\} \to \mathbb{N} \cup \{\infty, \emptyset\}$, such that $I(u, v) = I(u, x)$ for some $u \neq x \neq v \in V$ implies $I(u, v) = I(u, x) = \emptyset$, and $I(u, v) \in \mathbb{N}$ implies $I(u, v) \leq |\Gamma(u)|$. We say that a preference pattern L for a supergraph G' of G *agrees* with an incomplete preference pattern I if, for all $uv \in E(G)$ with $I(u, v) \neq \emptyset$, we have: $L(u, v) = I(u, v)$ whenever $I(u, v) \in \mathbb{N}$ and $L(u, v) = |\Gamma_{G'}(u)|$ whenever $I(u, v) = \infty$ (here $\Gamma_{G'}(u)$ is the neighborhood of u in G').

We discuss next some basic definitions and properties on graphs and matchings. For standard definitions that do not appear in here, we refer the reader to a textbook, such as [18]. As all graphs are simple, path and cycles are uniquely identified by the ordered list of their nodes, and represented as such. Fix a matching M of G. An *M-alternating path* (resp. *cycle*) in G is a path (resp. cycle) whose edges are alternatively in M and in $E(G) \setminus M$. To each edge $e = uv \in E \setminus M$, we associate an ordered pair of labels $(*_u, *_v)$ with $*_u, *_v \in \{+, -\}$ as follows: for $z \in \{u, v\}, x \in \{u, v\}$ and $z \neq x$, $*_z = +$ if $x >_z M(z)$ or z is unmatched, and $*_z = -$ otherwise. We write $uv = (+, +)$ or *uv is a $(+, +)$ edge* if $*_u = *_v = +$, and similarly for the other cases. Note that we are slightly abusing notations because edges are undirected pairs of nodes, while edge labels are a directed pair. However, this will never cause confusion in the following. The graph G_M is then defined as the subgraph of G obtained by deleting edges that are labeled $(-, -)$. Observe that M is also a

matching of G_M, hence definitions of M-alternating path and cycles apply in G_M as well. These definitions can be used to obtain a characterization of popular matchings in terms of forbidden substructures of G_M [38].

Theorem 13. *Let $(G, <)$ be a preference system, and M a matching of G. Then M is popular if and only if G_M does not contain any of the following:*

(i) an M-alternating cycle with a $(+,+)$ edge.

(ii) an M-alternating path that starts and ends with two distinct $(+,+)$ edges.

(iii) an M-alternating path starting from an M-exposed node and ending with a $(+,+)$ edge.

If the matching M under consideration is clear, we omit mentioning that labels are wrt M and paths, cycles etc. occur in the graph G_M: we simply say e.g. that uv is a $(+,+)$ edge and P is an M-alternating path with a $(+,+)$ edge. In particular, unless stated otherwise, when talking about M-alternating paths or cycles, we always mean in the graph G_M (hence, after $(-,-)$ edges have been removed).

We say a matching M *defeats* a matching M' (and M' is defeated by M) if either of these two conditions holds:

(i) M is more popular than M', i.e., $\phi(M, M') > \phi(M', M)$;

(ii) $\phi(M, M') = \phi(M', M)$ and $|M| > |M'|$.

A *dominant* matching is defined to be a matching that is never defeated. A dominant matching is by definition also popular, and among popular matching, is a matching of maximum size. The following characterization of dominant matchings appeared in [16], where it is also shown that a dominant matching always exists.

Theorem 14. *Let M be a popular matching. M is dominant if and only if there is no M-augmenting path in G_M.*

The following fact is attributed in [15] to [37]. We give a proof in Appendix B.1 for completeness.

Lemma 4.3.1. *Let $(G, <)$ be a preference system, and M (resp. S, D) be a popular matching (resp. popular matching of minimum size, popular matching of maximum size) in $(G, <)$. Then $V(S) \subseteq V(M) \subseteq V(D)$.*

For the positive results from Theorem 11 and Theorem 10 we will extensively use structural results and efficient algorithms from [16], which we will present when needed.

4.3.2 Graph theory

Let $G(V, E)$ be a graph. For $S \subseteq V$, we denote by $G \setminus S$ the subgraph of G induced by $V \setminus S$. If $G \setminus S$ is disconnected, S is said to be a (*vertex*) *separator*. Given a matching M on G and a set $U \subseteq V$, the matching induced by M on U is given by $E[U] \cap M$, and it is denoted by $M[U]$, while the subgraph of G induced by U, denoted by $G[U]$, is the graph $G(U, F)$ with $F = \{uv \in E : u, v \in U\}$.

More treewidth-specific concepts will be presented in Section 4.6.

4.4 The reduction

4.4.1 From a restricted 3-SAT formula to $G(\psi)$

A 3-SAT instance ψ is a logic formula given by the conjunction of a collection of $\mathcal{K}$ clauses, each of which is the disjunction of at most 3 literals. We denote the set of variables by $\mathcal{X} = \{x_1, \ldots, x_n\}$, the set of clauses by $C = \{c_1, \ldots, c_m\}$, and the set of literals by $\mathcal{L} = \{\ell_1, \ldots, \ell_p\}$, with $p \leq 3m$. Literals of the form x_i are called *positive* (*occurrences of variable i*), while those of the form $\neg x_i$ are called *negative* (*occurrences of variable i*). In this chapter, we restrict our attention to monotone $3-SAT$, which is the version of $3-SAT$ where each clause consists of only positive or only negative literals. This is also known to be NP-hard, see [8][1]. We refer to these

[1] A reduction from $3-SAT$ is obtaining replacing a clause $c = (x_1 \vee \neg x_2 \vee \neg x_3)$ by $(x_1 \vee x_c) \wedge (\neg x_c \vee \neg x_2 \vee \neg x_3)$ and a clause $c = (x_1 \vee x_2 \vee \neg x_3)$ by $(x_1 \vee x_2 \vee x_c) \wedge (\neg x_c \vee \neg x_3)$, where for each clause c, x_c is a new variable. The

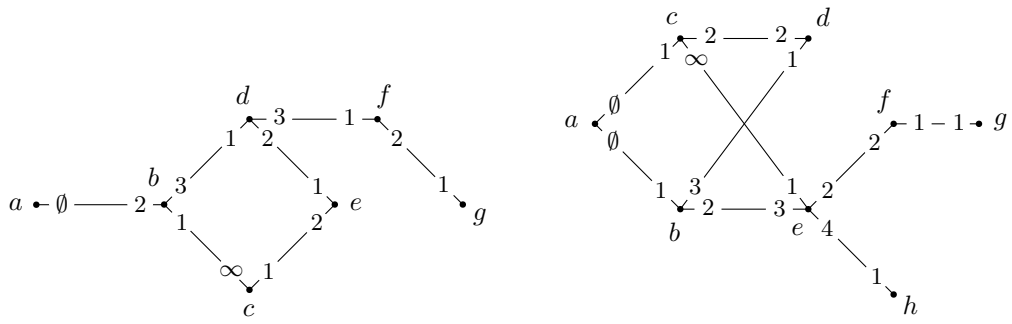

Figure 4.1: The gadget associated to the positive (left) and negative (right) literals, and the corresponding canonical incomplete preference patterns.

clauses as positive clauses and negative clauses accordingly. From now on, we assume without loss of generality that the monotone 3-SAT instances under consideration do not consist of only positive (resp. negative) clauses, and that no variable only appears as a positive (resp. negative) literal. Also, for a feasible assignment to ψ, we say that a positive (resp. negative) literal is true if the corresponding variable is set to true (resp. false), and that it is false otherwise.

Define the *positive gadget* to be the graph H constructed as in Figure 4.1, left. Node a is called the *apex* of H. Node c is called the *consistency enforcer* of H. Node g is called the *gateway* of H. Note that Figure 4.1, left, also defines an incomplete preference pattern I_H for H, which we call *canonical*.

Define the *negative gadget* to be the graph N constructed as in Figure 4.1, right. Node a is called the *apex* of N. Node c is called the *consistency enforcer* of N. Node g is called the *gateway* of N. Edge eh is called the *evicted* edge of N. Figure 4.1, right, also defines an incomplete preference pattern I_N for N, which again we call *canonical*.

To each positive (resp. negative) literal ℓ of a restricted 3−SAT instance, we associate one positive (resp. negative) gadget $H(\ell)$ (resp. $N(\ell)$). We will write $G(\ell) = (V(\ell), E(\ell))$ to denote the gadget associated to a literal ℓ, whether it is positive or negative. Nodes of $G(\ell)$ will be denoted by $a(\ell)$, $b(\ell)$, ...

Consider a monotone 3-SAT instance ψ. Fix an arbitrary order of clauses and of literals in each

other clauses remain unchanged.

clause. Define the graph $G := G(\psi)$ as follows. First, construct disjoint graphs $G(\ell)$ for each literal ℓ, and add nodes u and v. Now fix a clause c, and let ℓ_1, ℓ_2, ℓ_3 be the literals from the clause, in this order (with ℓ_3 possibly non-existent). Identify the apex of $G(\ell_1)$ with u; the apex of $G(\ell_i)$, $i \in [2,3]$ with the gateway of $G(\ell_{i-1})$; and add edge $g(\ell_k)v$, where ℓ_k is the last literal of the clause. Repeat the operation for all clauses (note that vertices u and v are unique). Let $\bar{G}(\bar{V}, \bar{E})$ be the graph obtained, and let

$$V(G) := \{s, t, w, x, y\} \cup \bar{V}.$$

Define $E(G) := \bar{E} \cup E_1 \cup E_2$, where:

$E_1 := \{st, tu, vw, wx, xy\}$;

$E_2 := \cup_{i=1,\ldots,n}\{c(\ell)c(\ell') : \ \ell$ (resp.ℓ') is a positive (resp. negative) occurrence of variable $i\}$.

We refer to edges E_2 as *consistency edges*. See Figure 4.2 for an example of the graph $G(\psi)$.

Lemma 4.4.1. *Let $G(\psi)$ be defined as above for a monotone $3-SAT$ instance ψ. Then $G(\psi)$ is bipartite.*

Proof. We provide a bipartition of $G(\psi)$ by assigning each vertex from $V := V(G(\psi))$ to exactly one of two sets A or B, and proving that if $u, v \in A$ (or $u, v \in B$) then $uv \notin E := E(G(\psi))$. The proof can be followed in Figure 4.2, where nodes are colored white (resp. black) if they belong to A (resp. B). Assign s, u, w, y to A and t, v, x to B. For any ℓ corresponding to a positive literal, assign $a(\ell), d(\ell), c(\ell), g(\ell)$ to A and $b(\ell), e(\ell), f(\ell)$ to B. For any ℓ corresponding to a negative literal, assign $a(\ell), d(\ell), e(\ell), g(\ell)$ to A and $b(\ell), c(\ell), f(\ell), h(\ell)$ to B. Note that this assignment is consistent with the fact that some $a(\ell)$ and $g(\ell)$ are identified, and that some $a(\ell)$ are identified with u (they are all assigned to A). One easily checks that edges in $\bar{E} \cup E_1$ connect nodes in the same set of the bipartition. Now consider $c(\ell)c(\ell') \in E_2$. By construction, we can assume wlog that ℓ (resp. ℓ') is a positive (resp. negative) literal. Hence $c(\ell) \in A$, $c(\ell') \in B$, as required. □

Our hardness and hardness of approximation proofs are based on the following lemma.

Lemma 4.4.2. *Consider the graph $G := G(\psi)$, and set $F := \{st, wx\}$. There exists a preference system $<$ with the following property: $(G(\psi), <)$ has a popular matching that does not contain any edge in F if and only if ψ is satisfiable. Given ψ, this preference system can be found in polynomial time.*

We devote the rest of Section 4.4 to the proof of Lemma 4.4.2. The reader interested in how Lemma 4.4.2 is applied to deduce the proofs of main results can skip directly to Section 4.5.

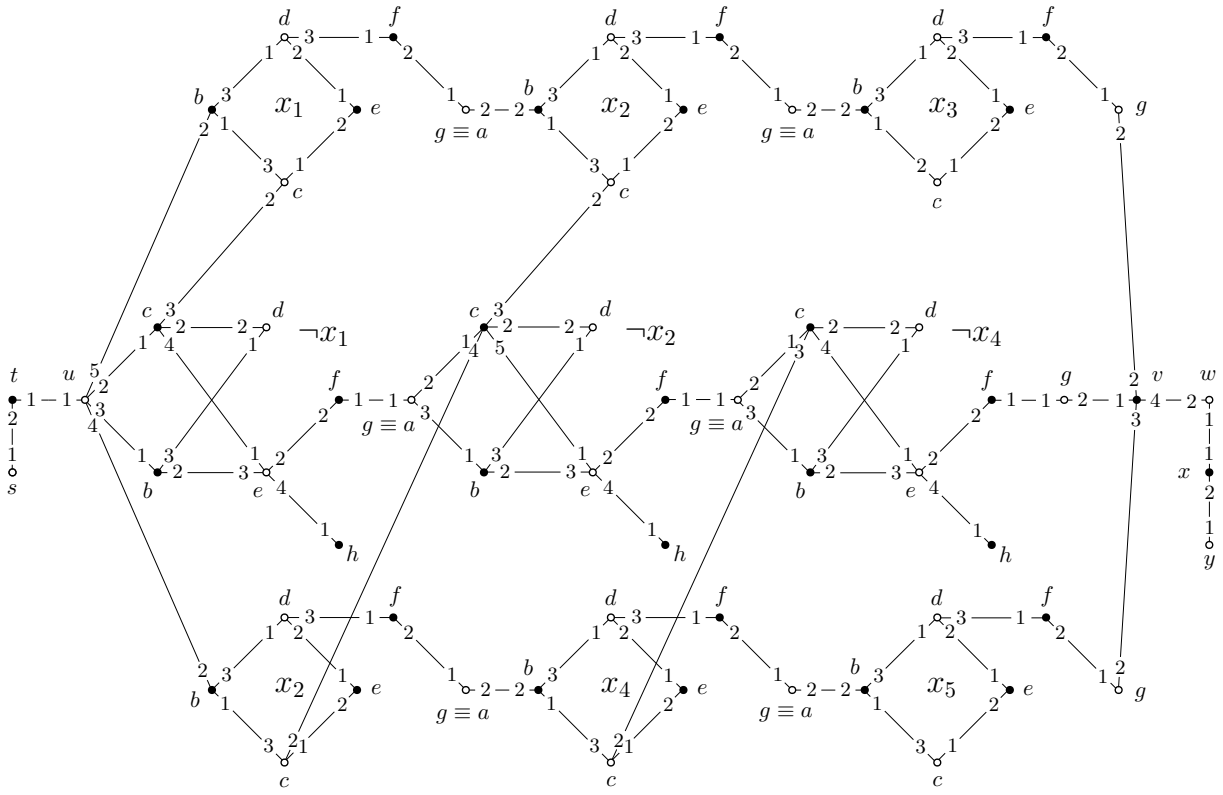

Figure 4.2: A monotone 3-SAT instance $\psi = (x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_4) \wedge (x_2 \vee x_4 \vee x_5)$ and the associated graph $G(\psi)$ and preference system. Disk and circle labels of vertices provide a bipartition of $G(\psi)$.

4.4.2 The preference system

In the remainder of Section 4.4, we fix a monotone 3-SAT instance ψ and the corresponding graph $G(V, E) := G(\psi)$. We start by showing how to define the preference system via the preference pattern L.

Consider the union I of all canonical incomplete preference patterns for gadgets corresponding to literals of ψ. Recall that vertex sets of gadgets are disjoint subsets of V, with the exception of $g(\ell_i) \equiv a(\ell_{i+1})$, with ℓ_{i+1} being the literal following ℓ_i in some clause, and $u \equiv a(\ell_1)$, where ℓ_1 is the first literal in some clause. However, the canonical incomplete preference patterns assign $\emptyset$ to edges incident to nodes $a(\ell)$. Hence, I is an incomplete preference pattern for G.

Consider now the following preference pattern L for G. First, L agrees with I. Next, we have $L(s,t) = 1$, $L(t,s) = 2$, $L(u,t) = L(t,u) = 1$, $L(v,w) = |\Gamma(v)|$, $L(w,v) = 2$, $L(w,x) = 1$, $L(x,w) = 1$, $L(x,y) = 2$, $L(y,x) = 1$. This fixes $L(\cdot)$, with the exception of the following pairs:

- $L(u, b(\ell))$, for all positive literals ℓ that are first in some clause, and $uc(\ell)$ for all negative literals that are first in some clause. Since $L(u,t) = 1$ and no other edge is incident to u, we can assign $L(u,v)$ for $v = b(\ell)$ or $v \in \{b(\ell), c(\ell)\}$ as above via an arbitrary bijective map to $\{2, \ldots, |\Gamma(v)|\}$.

- $L(g(\ell), v)$ and $L(v, g(\ell))$, for all literals ℓ that are last in some clause. Since all such $g(\ell)$ have degree 2 in G, and $L(g(\ell), f(\ell)) = 1$, we set $L(g(\ell), v) = 2$. Moreover, $\Gamma(v)$ is composed of all such $g(\ell)$ and vertex w, to which we already assigned $L(v,w) = |\Gamma(v)|$. Hence, we assign values $L(v, g(\ell))$ for ℓ as above via an arbitrary bijective map to $\{1, \ldots, |\Gamma(v)| - 1\}$.

- $L(c(\ell), c(\ell'))$ and $L(c(\ell'), c(\ell))$, where ℓ is a positive occurrence and ℓ' is a negative occurrence of the same variable. Note that we already assigned $L(c(\ell), e(\ell)) = 1$, $L(c(\ell), b(\ell)) = |\Gamma(c(\ell))|$. We assign $L(c(\ell), c(\ell'))$ for a fixed $c(\ell)$ via an arbitrary bijective map to $\{2, \ldots, |\Gamma(c(\ell))| - 1\}$. Similarly, we have already assigned $L(c(\ell'), a(\ell')) = 1$, $L(c(\ell'), d(\ell')) = 2$, $L(c(\ell'), e(\ell')) = |\Gamma(c(\ell'))|$. Hence, we assign $L(c(\ell'), c(\ell))$ for a fixed $c(\ell')$ via an arbitrary bijective map to $\{3, \ldots, |\Gamma(c(\ell'))| - 1\}$

- $L(a(\ell), b(\ell))$ for all positive literals ℓ that are not first in some clause. Recall that we have $L(a(\ell), f(\ell')) = 1$, where ℓ' is the literal preceding ℓ in its clause. We set therefore $L(a(\ell), b(\ell)) = 2$.

- $L(a(\ell), b(\ell))$, $L(a(\ell), c(\ell))$ for all negative literals ℓ that are not first in some clause. Recall that we have $L(a(\ell), f(\ell')) = 1$, where ℓ' is the literal that precedes ℓ in the clause. We set therefore $L(a(\ell), b(\ell)) = 3$, $L(a(\ell), c(\ell)) = 2$.

We let $<$ be the preference system corresponding to the preference pattern L. We now derive properties of matchings that are popular in $(G, <)$ and satisfy some further hypothesis. In the proof of the next and following lemmas, we often deduce a contradiction using the characterization of popular matchings given in Theorem 13, hence we omit referring explicitly to this theorem each time.

4.4.3 Properties of certain popular matchings of $(G, <)$

Lemma 4.4.3. *Let M be a popular matching of $(G, <)$ that does not contain any evicted edge. Then M does not contain any consistency edge, i.e., $M \cap E_2 = \emptyset$.*

Proof. Suppose by contradiction that M contains a consistency edge, connecting a node from the positive gadget $H(\ell)$ to a node from the negative gadget $N(\ell')$. We focus now on nodes of N, hence omit the dependency on ℓ'. Since the consistency edge incident to c is matched, if d is not matched, then cd would be a $(+,+)$ edge adjacent to an unmatched vertex, which contradicts popularity. Hence, $bd \in M$, which in turn implies $ef \in M$, otherwise be is a $(+,+)$ edge adjacent to an unmatched vertex, again a contradiction (recall that $eh \notin M$ by hypothesis). This implies that eh is a $(-,+)$ edge, fg is a $(+,+)$ edge, and h, e, f, g is an M-alternating path containing a $(+,+)$ edge, a contradiction. □

Lemma 4.4.4. *Let M be a popular matching of $(G, <)$ that contains tu. Then:*

1. *The gateway (resp. apex) of each gadget is matched to its favorite partner;*
2. *M does not contain any consistency or evicted edge;*
3. *In each positive gadget $H(\ell)$, exactly one of the following is true (dependency on ℓ is omitted):*

(a) $\mathcal{F}(H) := \{bd, ce\} \subseteq M$, *or*

(b) $\mathcal{T}(H) := \{bc, de\} \subseteq M$.

Similarly, in each negative gadget $N(\ell')$*, exactly one of the following is true (dependency on* ℓ' *is omitted):*

(a) $\mathcal{F}(N) := \{cd, be\} \subseteq M$, *or*

(b) $\mathcal{T}(N) := \{ce, bd\} \subseteq M$.

4. *If* ℓ *is a positive and* ℓ' *a negative occurrence of the same variable, then we cannot have both* $\mathcal{T}(H(\ell)) \subseteq M$ *and* $\mathcal{T}(N(\ell)) \subseteq M$.

Proof. 1. We consider negative and positive gadgets separately. Note that it is enough to prove the statement for the gateway, since the one for the apex follows by the fact that each apex is either also a gateway, or u (that is matched to its favorite partner t by hypothesis).

Pick first a negative clause and one of its literals ℓ. We omit the dependency on ℓ in nodes of $N(\ell)$. The favorite partner of g is f. Hence, suppose by contradiction that $fg \notin M$. Note that since g is also f's favorite partner, fg is a $(+,+)$ edge, which implies $fe \in M$, else f is an unmatched node adjacent to a $(+,+)$ edge, and we conclude that e, f, g is an M-alternating path starting at an unmatched node that contains a $(+,+)$ edge, a contradiction. Edge eh is therefore labeled $(-,+)$, and h is unmatched, since its only neighbor is e. The path h, e, f, g is therefore an M-alternating path starting from an unmatched node that contains a $(+,+)$ edge, a contradiction. This concludes the proof for this case.

Consider any positive clause, and take the first of its literals whose getaway is not matched to its favorite partner. Call this literal ℓ. Again, we omit the dependency on ℓ in nodes of $H(\ell)$, and observe that f is g's favorite partner. Therefore, $fd \in M$, else f is unmatched and fg is a $(+,+)$ edge adjacent to an unmatched node, a contradiction. If e is unmatched, then de is a $(+,+)$ edge adjacent to an unmatched node, again a contradiction. Since $\Gamma(e) = \{d, c\}$ and $df \in M$, we conclude $ec \in M$. Similarly, if b is unmatched, then bd is $(+,+)$ edge, hence $ab \in M$. Now,

if ℓ is the first literal of the clause, $a \equiv u$, contradicting $tu \in M$. Else, $a \equiv g(\ell')$ for the literal ℓ' preceding ℓ in the clause, and since ℓ was the first literal in the clause that violated the thesis, $f(\ell')g(\ell') \in M$, a contradiction.

2. Suppose by contradiction M contains an evicted edge. Pick any clause with a negative literal ℓ such that $c(\ell)c(\ell') \in M$ for some ℓ'. We omit the dependency on ℓ in nodes of $N(\ell)$. Part 1 implies that a is matched to its favorite partner. The hypothesis implies that $eh \in M$. Hence, if $bd \notin M$, then be is a $(+,+)$ edge incident to the unmatched node b, a contradiction. Hence $bd \in M$. Since a, d, e are not matched to c in M, if c is not matched through a consistency edge it is unmatched. Then ce is a $(+,+)$ edge adjacent to an unmatched node, a contradiction. A similar argument implies $d(\ell')e(\ell') \in M$ and $a(\ell')b(\ell') \in M$, contradicting part 1. This concludes the proof that M does not contain any evicted edge. The fact that M does not contain any consistency edge immediately follows from what just proved and Lemma 4.4.3.

3. Fix a gadget $G(\ell)$ and omit dependency on ℓ in nodes (recall that we denote by $G(\ell)$ a generic gadget, that could be positive or negative). By part 1 and 2, a is matched to its first choice, and M does not contain any consistency or evicted edges. Hence, nodes b, c, d, e are either matched to each other, or unmatched. Simple case checking shows that, if any of b, c, d, e is unmatched, then there is a $(+,+)$ edge incident to an unmatched node, a contradiction to popularity. Hence, each of these vertices must be matched, and since they form a cycle with 4 vertices, one of the possibilities from the thesis of part 3 of the lemma must hold.

4. Suppose we have both $\mathcal{T}(H(\ell)) \subseteq M$ and $\mathcal{T}(N(\ell')) \subseteq M$. Then the consistency edge $c(\ell)c(\ell')$ is a $(+,+)$ edge. Hence, $h(\ell'), e(\ell'), c(\ell'), c(\ell)$ is an M-alternating path from an unmatched node to a $(+,+)$ edge, a contradiction. □

4.4.4 From popular matchings to feasible assignments

Let M be a popular matching M of $(G, <)$ that does not contain edges in $F = \{st, wx\}$. In order to avoid one of the forbidden configurations from Theorem 13, we deduce that $tu, xy, vw \in M$. Observe that wx is a $(+,+)$ edge and s is unmatched.

We claim that each clause contains at least one literal ℓ with $T(\ell) \subseteq M$. We show that, if this does not happen, then there exists an M-alternating path from s to wx. Together with the fact that wx is a $(+,+)$ edge and s is unmatched, this contradicts the popularity of M. The construction of this path can be followed in Figure 4.3. We assume that the clause is given by exactly three positive literals. Similar arguments apply for the case with two positive literals. Recall that, by Lemma 4.4.2, part 1, apexes and gateways are matched to their favourite partner.

Suppose first we deal with a positive clause, and let $H(\ell_1), H(\ell_2), H(\ell_3)$ be the corresponding gadgets. By Lemma 4.4.4, part 3, $\mathcal{F}(\ell_1), \mathcal{F}(\ell_2), \mathcal{F}(\ell_3) \subseteq M$. Suppose first $H(\ell)$ is a positive gadget. Then st, $a(\ell)b(\ell)$, $d(\ell)f(\ell)$, $g(\ell_3)v$ are $(+,-)$ or $(-,+)$ edges for $\ell = \ell_1, \ell_2, \ell_3$, and

$$s, t, u, b(\ell_1), d(\ell_1), f(\ell_1), g(\ell_1), b(\ell_2), d(\ell_2), f(\ell_2), g(\ell_2), b(\ell_3), d(\ell_3), f(\ell_3), g(\ell_3), v, w, x$$

is the promised M-alernating path.

Consider now a clause that is given by three negative literals: $N(\ell_1), N(\ell_2), N(\ell_3)$. Then st, $a(\ell)b(\ell)$, $e(\ell)f(\ell)$, $g(\ell_3)v$ are $(+,-)$ or $(-,+)$ edges for $\ell = \ell_1, \ell_2, \ell_3$, and the the alternating path is

$$s, t, u, b(\ell_1), e(\ell_1), f(\ell_1), g(\ell_1), b(\ell_2), e(\ell_2), f(\ell_2), g(\ell_2), b(\ell_3), e(\ell_3), f(\ell_3), g(\ell_3), v, w, x, y.$$

From M, we obtain an assignment of the literals for $3-SAT$ as follows: if $\mathcal{T}(\ell) \subseteq M$ for some positive (resp. negative) occurrence ℓ of variable x_i, then set x_i to be true (resp. false). Observe first that the assignment is well-defined, since we cannot assign the same variable to simultaneously true and false, because of Lemma 4.4.4, part 4. However, notice that although positive and negative gadgets can not both be true, they can both be false. Since by the first part of the proof, in each clause there is at least an ℓ such that $T(\ell) \subseteq M$, we conclude that ψ is satisfied by the assignment we constructed. Hence, if there is a popular matching that does not contain any edge from F, then ψ is satisfiable.

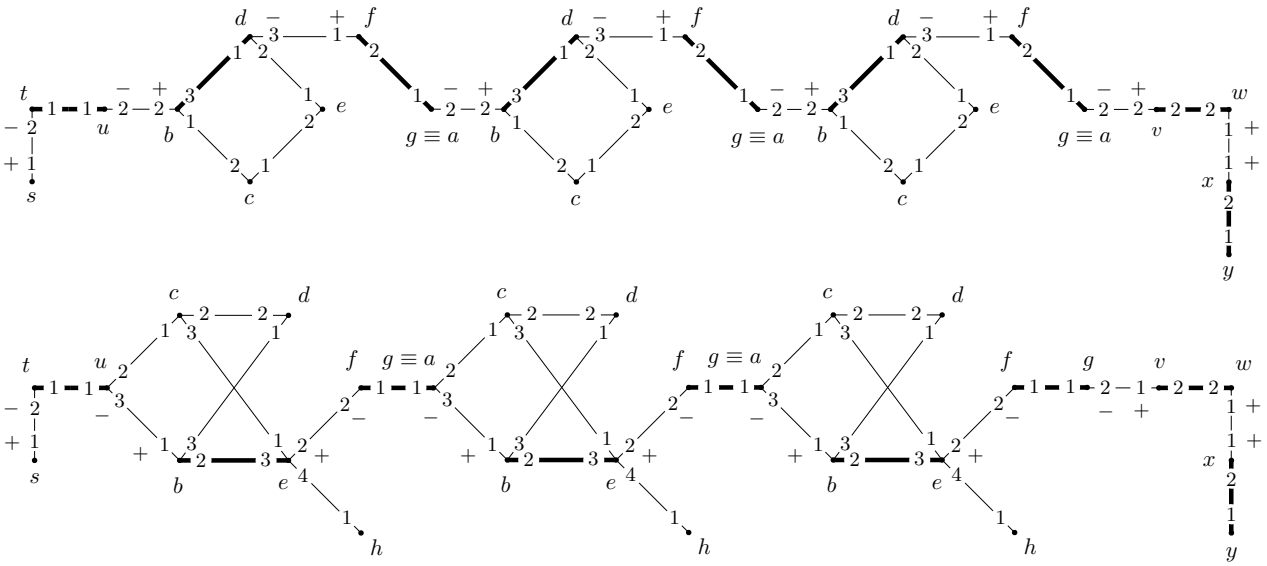

Figure 4.3: An alternating path from s to w when all literals are False in positive (above) and negative (below) clauses.

4.4.5 From feasible assignments to popular matchings

Now suppose that we have an assignment of the literals which satisfies ψ. Construct a matching M as follows. First, add edges tu, vw, xy, and edges $f(\ell)g(\ell)$ for each literal ℓ. Then, for each positive literal ℓ, add $\mathcal{T}(\ell)$ to M if the corresponding variable is set to true; add $\mathcal{F}(\ell)$ to M if the corresponding variable is set to false. Last, for each negative literal ℓ, add $\mathcal{T}(\ell)$ to M if the corresponding variable is set to false; add $\mathcal{F}(\ell)$ to M if the corresponding variable is set to true. Figure 4.4 gives an example of a matching M corresponding to a feasible assignment of the clause from Figure 4.2. Note that M does not contain st and wx, hence it does not contain edges in F.

We now show that M constructed above is popular by proving it satisfies the conditions of Theorem 13. This concludes the proof of Lemma 4.4.2. The following claim characterizes the only $(+,+)$ edge.

Claim 4.4.5. *The only edge of $E \setminus M$ labelled $(+,+)$ is wx.*

Proof of claim. By construction $tu, vw, xy \in M$. Consider next a consistency edge $c(\ell)c(\ell')$, which connects a positive gadget $H(\ell)$ and a negative gadget $N(\ell')$. By construction, the following

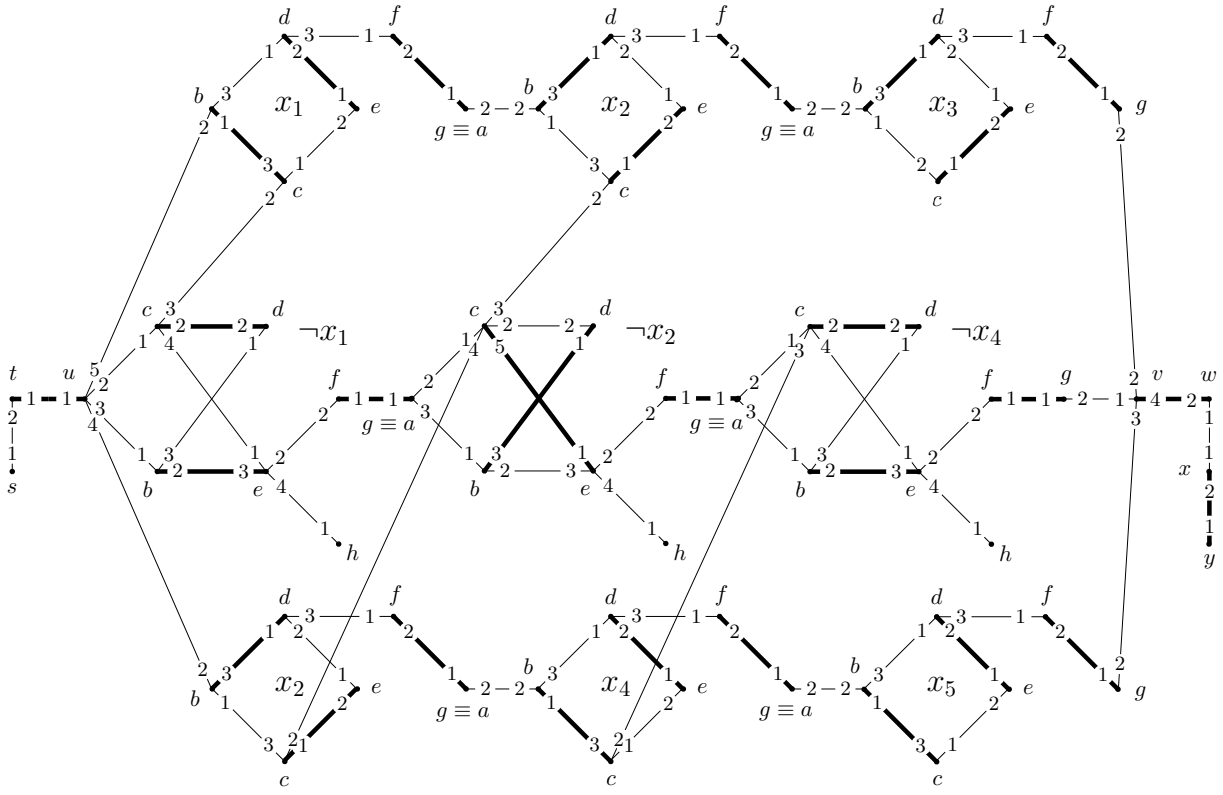

Figure 4.4: A matching M (bold edges) corresponding to a feasible assignment of $\psi = (x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_4) \wedge (x_2 \vee x_4 \vee x_5)$ from Figure 4.2: x_1, x_4, x_5 are set to True and x_2, x_3 to False.

two cases happen: $\mathcal{T}(H(\ell))$, $\mathcal{N}(H(\ell') \subseteq M$, or $\mathcal{T}(H(\ell'))$, $\mathcal{N}(H(\ell) \subseteq M$. One easily checks that therefore $c(\ell)c(\ell') = (-,+)$ or $c(\ell)c(\ell') = (+,-)$.

Notice that $st = (+,-)$, and all edges of $E \setminus M$ incident on a apex or a gateway have at least one label equal to "−" since, by construction, apexes and gateways are matched to their favourite partners. Hence, we are left with edges between nodes b, d, c, e, f and possibly h (in negative literals) of the same gadget $G(\ell)$ (we omit dependency on ℓ in nodes). Suppose first ℓ is a positive literal. If ℓ is true, then b, e are matched to their favorite partners, hence no edge incident to b, e can be a $(+,+)$ edge. If ℓ is false, c, d are matched to their favorite partners, and no edges incident to c, d can be a $(+,+)$ edge. Moreover, in all cases, df is a $(-,+)$ edge. This shows that no edge incident to a node of a positive gadget can be labelled $(+,+)$. Suppose now that ℓ is a negative literal. If ℓ is true, then e, c are matched to their favorite partners, concluding the proof. Else, $cd, be \in M$, and then db, ce, eh, fe are $(-,+)$ edges, concluding the proof. ⋄

Claim 4.4.5 implies that conditions (i) and (ii) of Theorem 13 are satisfied by M. Hence, we are left to show that there is no M-alternating path P that starts from an unmatched node and contains a $(+,+)$ edge. Call such a path *malicious*. Observe that, by construction, the only unmatched nodes are s, $h(\ell)$ for all negative literals ℓ, and that any malicious path contains wx because of Claim 4.4.5. The proofs of next claims are similar to those of previous statements, and are therefore postponed to Appendix B.2.

Claim 4.4.6. *If a positive clause contains a true literal ℓ, then $a(\ell), b(\ell) = (-,-)$. If a negative clause contains a true literal ℓ', then $e(\ell')f(\ell') = (-,-)$.*

Claim 4.4.7. *For any clause c, there is no malicious path starting at s that is contained in $\{s,t,u,v,w,x\} \cup \bigcup_{\ell \in c} V(\ell)$.*

Claim 4.4.8. *A malicious path starting at $h(\ell)$ does not traverse $f(\ell)$. In particular, there is no malicious path from $h(\ell)$ to wx that is contained in $\{s,t,u,v,w,x\} \cup \bigcup_{\ell \in c} V(\ell)$ for any negative clause c.*

Claim 4.4.9. *Let ℓ be a positive literal, and let P be a malicious path. Then $c(\ell)c(\ell')$ is an edge of P for at most one negative literal ℓ' and if that happens, then $f(\ell) \notin P$.*

Claim 4.4.10. *Let ℓ be a negative literal, and let P be a malicious path. Then $c(\ell)c(\ell')$ is an edge of P for at most one ℓ' and if that happens and $a(\ell) \neq u$, then $a(\ell) \notin P$.*

In order to conclude that condition (iii) of Theorem 13 is satisfied by M, suppose first, by contradiction, there exists a malicious path P starting from s. Then $P = s, t, u, \ldots$. Claim B.2.2 implies that there is at least one consistency edge that belongs to P. Take the first consistency edge $c(\ell)c(\ell')$ that P traverses, and suppose without loss of generality that $c(\ell)$ is traversed by P before $c(\ell')$. This means that $a(\ell)$ and z are two consecutive nodes of P, traversed in this order and before $c(\ell)$, for some $z \in V(\ell)$ (possibly $z = c(\ell)$). Suppose first ℓ is a positive literal. Hence, $a(\ell), b(\ell) \in P$. By Claim B.2.1, ℓ is false, hence ℓ' is true. Using again Claim B.2.1, we deduce $f(\ell') \notin P$. Hence P, in order to reach wx, must traverse $a(\ell')$ after $c(\ell')$. If $a(\ell') = u$, then P cannot traverse u again. Hence $a(\ell') \neq u$ and, by Claim B.2.5, $a(\ell') \notin P$. In both cases, P cannot reach wx, a contradiction. If conversely ℓ is a negative literal, by Claim B.2.5 $a = u(\ell)$. In the following, we omit dependency on ℓ in nodes of $H(\ell)$. If ℓ is set to true, then by construction $cd, be \in M$. Hence, P contains the subpath u, b, c, d, e or the subpath u, c, e. In both cases, $c(\ell)c(\ell')$ cannot be an edge of P, a contradiction. If conversely ℓ is set to false, then P contains the subpath u, b, e or u, c, d . In the latter case, $cc(\ell')$ cannot be an edge of P. In the former, the inverse of P must contain the subpath $c(\ell'), c, d, b$, a contradiction (b would have degree 3 in P). Hence, we can conclude that there is no malicious path starting from s.

Suppose there exists a malicious path P starting from $h(\ell'')$, with ℓ'' being a negative literal. Since $tu \in M$, $u \notin P$. From Claim B.2.3, P must traverse a consistency edge $c(\ell')c(\ell)$ with ℓ' being a false literal and $c(\ell')$ preceding ℓ in the path. Pick the first such edge. By Claim B.2.4, $f(\ell) \notin P$. Hence, after traversing $c(\ell)$, P visits only gadgets corresponding to literals preceding ℓ in the clause, until it traverses (in this order) another consistency edge $c(\ell''')c(\ell'''')$, since as we argued above, $u \notin P$. This contradicts Claim B.2.4, since it must also be that $f(\ell''') \in P$.

Since s and the $h(\ell)$ with ℓ being a negative literal are the only unmatched vertices, we conclude

that there is no malicious path. Hence, condition (*iii*) is also satisfied, and M is popular as claimed.

4.5 Consequences of the reduction

4.5.1 Proof of Theorem 10

Proof. Consider the graph $G(\psi)$ from Lemma 4.4.2, and give weight 1 to edges ut, xy, and 0 to all other edges. Suppose we want to solve `mppb`. Because of Lemma 4.4.2, it is NP-Complete to decide whether $G(\psi)$ has a popular matching of weight 2. In order to prove the positive result, consider the following claim.

Claim 4.5.1. *Let M^*_{st} and M^*_{dom} be the maximum weight stable and dominant matchings of (G, ψ) respectively. Then $w(M) \leq 2W$ for any popular matching M, where $W = \max(w(M^*_{st}), w(M^*_{dom}))$.*

Proof of claim. Assume by contradiction that there exists a popular matching M such that $w(M) > 2W$. It was shown in [16] that any popular matching M can be partitioned into a stable matching M_0 restricted to a subgraph $G'(\psi)$ and a dominant matching $M_1 = M \setminus M_0$ restricted to $G(\psi) \setminus G'(\psi)$, and there always exist matchings M'_0 and M'_1 such that $M'_{st} = M_0 \cup M'_1$ is stable and $M'_{dom} = M'_0 \cup M_1$ is dominant in $G(\psi)$. First assume $w(M_0) \geq w(M_1)$. This implies $w(M_0) = \frac{1}{2}(w(M_0) + w(M_0)) \geq \frac{1}{2}(w(M_0) + w(M_1)) = \frac{1}{2}w(M)$. Consider $M'_{st} = M_0 \cup M'_1$, $w(M'_{st}) = w(M_0) + w(M'_1)$. Since weights are non-negative, $w(M'_{st}) \geq w(M_0) \geq \frac{1}{2}w(M)$. We conclude that $W \geq w(M'_{st}) \geq \frac{1}{2}w(M)$. This contradicts to the assumption that $w(M) > 2W$. If $w(M_1) \geq w(M_0)$, apply the same argument to M'_{dom}. ◇

The 1/2-approximation immediately follows from the previous claim, and the fact that a dominant and stable matching of maximum weight can be computed in polynomial time, see again [16].

Now change weights so that $w_e = -1$ if $e \in \{st, wx\}$, and $w_e = 0$ otherwise, and suppose we want to solve `mwpb` on the given instance. We deduce from Lemma 4.4.2 that it is NP-Complete to decide if the optimal solution is 0. Hence, the problem cannot be approximated to any factor, unless P=NP. □

4.5.2 Proof of Theorem 11

Proof. Suppose first $U^{in} = F^{in} = F^{out} = \emptyset$. Then the problem boils down to deciding if there exists a popular matching that does not contain a given set of nodes. By Lemma 4.3.1, it is enough to check if this is true for a(ny) stable matchings in $(G, <)$. It is well-known that this can be done in polynomial time, see e.g. [31].

Suppose now $U^{out} = F^{in} = F^{out} = \emptyset$. Then, again by Lemma 4.3.1, the problem corresponds to deciding if there exists a dominant matching of $(G, <)$ that contains a given set of nodes. It is shown in [16] how to efficiently construct an instance $(G', <')$ and a surjective map σ from stable matchings of $(G', <')$ to dominant matchings of $(G, <)$, with the additional property that a node $v \in V(G)$ is in a dominant matching $\sigma(S)$ of $(G, <)$ if and only if one of certain nodes v', v'' are in the stable matching S of $(G', <')$ (see [16] for details). Since, as mentioned above, whether a node belong to a(ny) stable matching can be checked efficiently, our problem can be solved in polynomial time.

Suppose now $U^{out} = U^{in} = F^{out} = \emptyset$, and $F^{in} = \{e\}$. This is the *popular edge problem*, that is shown in [16] to be solvable in polynomial time. We call the case when $U^{out} = U^{in} = F^{in} = \emptyset$, and $F^{out} = \{e\}$ the *ungusfield1989stablepular edge problem*. We show that a solution to the latter follows from the solution to cseh2018popular [16]. Define E_s and $\bar{E_s}$ as:

$$E_s = \{e \in E : \exists \text{ stable matching } \mathcal{M} \text{ s.t. } e \in \mathcal{M}\}$$

$$\bar{E_s} = \{e \in E : \exists \text{ stable matching } \mathcal{M} \text{ s.t. } e \notin \mathcal{M}\}$$

E_d, $\bar{E_d}$ (resp. E_p, $\bar{E_p}$) are defined similarly, by replacing "stable" with "dominant" (resp. popular). It is proved in [16] that $E_p = E_s \cup E_d$. We now argue that $\bar{E_p} = \bar{E_s} \cup \bar{E_d}$.

Consider any $e \in \bar{E_s} \cup \bar{E_d}$. Since there is a stable or dominant matching that does not contain e, it follows that $e \in \bar{E_p}$. We conclude that $\bar{E_p} \supseteq \bar{E_s} \cup \bar{E_d}$. Consider any $e = ij \in \bar{E_p}$. There is a popular matching P s.t. $e \notin P$, and i or j is matched (if i and j are unmatched then $(i, j) = (+, +)$, and M is not popular). Wlog assume $jk \in M$. It follows from $E_p = E_s \cup E_d$ that there exists a

stable or dominant matching M' s.t. $jk \in M'$, hence $ij \notin M'$. We conclude that $\bar{E_p} \subseteq \bar{E_s} \cup \bar{E_d}$.

Since $\bar{E_p} = \bar{E_s} \cup \bar{E_d}$, we can solve the forbidden edge problem by checking if $e \in \bar{E_s}$ or $e \in \bar{E_d}$. For stable matchings, this can be done in polynomial time, see e.g. [31]. For dominant matchings, this can be done thanks to the mapping σ define above, see [16] for details.

We now move to NP-Complete cases. Testing whether a matching is popular can be performed in polynomial time, see [6]. Hence all those problem are in NP. Note that, for a popular matching M of $G(\psi)$, the following are equivalent: $st, wx \notin M$; $tu, xy \in M$; $s \notin V(M)$ and $y \in V(M)$; $s \notin V(M)$ and $yx \in M$; $y \in V(M)$ and $st \notin M$; $y \in V(M)$ and $tu \in M$. Hence, Lemma 4.4.2 settles all NP-Complete cases with $|U^{in} \cup U^{out} \cup F^{in} \cup F^{out}| = 2$. To show NP-Completeness when sets U^{out} and/or F^{out} have bigger size, we can add nodes $h(\ell)$ to U^{out} and/or evicted edges to F^{out}, since we know from Lemma 4.4.4 that those nodes and edges are never in a popular matching that contains tu. If U^{in} and/or F^{in} have bigger sizes, we can repeatedly add nodes z, k, with z adjacent to k and k adjacent to $g(\ell)$ for some literal ℓ such that z, k are each other's favorite partner. Then clearly zk belongs to any popular matching, so it can be added to F^{in}, and z to U^{in}. □

4.5.3 Further results

We present here a couple of related results that easily follow from what presented so far. Next corollary deals with weighted popular matching problems where weights are given on the *nodes* instead of the edges.

Corollary 4.5.2. *The following problems:* **Given**: a preference system $(G, <)$ with weights w on nodes **Find**: a popular matching M that maximizes (resp. minimizes) $w(M)$ *are NP-Hard and not approximable to any factor in polynomial time, unless P=NP. Conversely, if* $w \geq 0$*, they can be solved in polynomial time.*

Proof. Note that, if we do not make any assumption on w, maximizing and minimizing are polynomially equivalent. So we focus on the maximization. Consider the instance $(G, <)$ as defined in Lemma 4.4.2, and w being the vector with -1 in the component corresponding to s, 1 in the

components corresponding to y, and 0 otherwise. Lemma 4.4.2 implies that it is NP-Complete to decide if the optimum is strictly greater than 0. On the other hand, if $w \geq 0$, Lemma 4.3.1 implies that the minimum (resp. maximum) of $w(M)$ with M popular is achieved at any stable (resp. dominant) matching, which can be easily computed. □

Recall that testing whether a matching is popular can be performed in polynomial time, see e.g. [6]. We conclude this section by showing that, on the other hand, deciding whether a subset of nodes can be matched among themselves only as to form a popular matching of the original instance is NP-Complete. We call this the *exclusive popular set problem.* It can also be seen as an extension of `pmfee` when U^{in} (or U^{out}) is the only non-empty set.

Corollary 4.5.3. *The following* exclusive popular set problem *(`eps`):* **Given**: a preference system $(G, <)$ and a set $U \subseteq V(G)$ **Decide**: if there exists a popular matching M of $(G, <)$ with $V(M) = U$ *is NP-Complete.*

Proof. Clearly the problem is in NP. Consider the instance from Lemma 4.4.2, and set $U = (\{t, u, v, w, x, y\} \cup_\ell V(\ell)) \setminus \cup_\ell \{h(\ell)\}$, where the union ranges over all literals. Let M be a popular matching of $(G, <)$. We claim that $V(M) = U$ if and only if $st, wx \notin M$. The thesis then follows by Lemma 4.4.2. Let $V(M) = U$. Then by definition $st \notin M$, and $wx \notin M$, otherwise y is unmatched. Now suppose $st, wx \notin M$. Then $tu, xy, vw \in M$, else M is not popular. By Lemma 4.4.4, we know that for each literal ℓ, $g(\ell)f(\ell) \in M$, $h(\ell)$ is M-exposed, and $c(\ell)c(\ell') \notin M$ for all ℓ'. Lemma 4.4.4, part 3, implies that, for a fixed ℓ, nodes $b(\ell), c(\ell), d(\ell), e(\ell)$ must be all matched, and all among themselves. This implies $V(M) = U$, as required. □

4.6 Popular matchings of maximum weights in graphs of bounded treewidth

In this section, we prove Theorem 12. The section is organized as follows. We start by giving an overview of the obstacles and of our techniques in Section 4.6.1. In Section 4.6.2, we recall some concepts and facts related to treewidth decompositions of graphs. In Section 4.6.3, we introduce the definitions of *locally popular matchings*, *configurations*, and *tipping points*, and show how the

partition a superclass of popular matchings in a small number of families. In Section 4.6.4, we show how to use those concepts together with a tree decomposition of bounded width to conclude the proof of Theorem 12.

4.6.1 An example and a high-level view of the algorithm

Before delving into the details of the algorithm, it is a useful exercise to investigate how `mwp` is different from certain problems in combinatorial optimization that are (NP-)hard in general, but can be solved efficiently on graphs of bounded treewidth. A classical such example is the Maximum Independent (Stable) Set (`MIS`) [7]. `MIS` enjoys two nice properties. The first is *monotonicity*: if S is an independent set in a graph G and G' is a subgraph of G, then the solution induced by S on G' is also feasible. The second is *locality*: S is an independent set if and only if, for each node $v \in S$, S does not intersect the neighborhood of v. These properties allow for an an optimal solution to `MIS` to be constructed iteratively as follows: start from a collection of "best" stable sets (with respect to the weight function) of a certain subgraph G', and "enlarge" them to a similar collections of stable sets in a larger subgraph G'' by looping over all sets of nodes from $V(G'') \setminus V(G')$ that can be potentially added. Roughly speaking, because of monotonicity, the restriction $S \cap V(G')$ of an optimal stable set S to vertices of G' is also a stable set, hence it is considered in the construction above. Moreover, locality implies that all the "best" stable sets stored for G' can be extended to stable sets of G.

Similar monotonicity and locality properties do not hold for popular matchings. Indeed, popularity is not a local condition, since it may depend on how nodes far away in the graph are matched, see Theorem 13. Moreover, if M is a popular matching in a graph G, the subset of M contained in an induced subgraph of G need not be popular. We will see an example of the latter fact, as well as our approach to bypass this issue, in Example 2 below. More generally, it does not seem easy to formulate `mwp` in *Monadic second order logic*, which is the most powerful general technique for obtaining linear-time algorithms in graphs of bounded treewidth, see [14].

Example 2. *Consider the roommate instance* $(G, <)$ *in Figure 4.5, top left. Matching* M *in Fig-*

ure 4.5, top right is popular in $(G, <)$*. However, the matching* $\overline{M}$ *induced by* M *on the subinstance* $(G', <)$ *of* $(G, <)$ *from Figure 4.5, bottom left, is not popular. Conversely, one readily checks that matching* M' *from Figure 4.5, bottom right, is popular in* $(G', <')$ *but cannot be "extended" to (i.e., it is not contained in) any popular matching of* $(G, <)$*.*

Let us now investigate popular matchings in $(G, <)$ *by first decomposing* G *into subgraphs* G' *and* G''*, and then finding matchings on* G' *that have the potential to be extended to a popular matching on the whole graph. In this example,* G' *and* G'' *are connected through a single vertex* g *(i.e.,* $\{g\}$ *is a vertex separator). This means that* G' *can only "influence"* G'' *through the vertex* g*. Having a separator of small size will limit the amount of information we need to store, and keep the algorithm polynomial.*

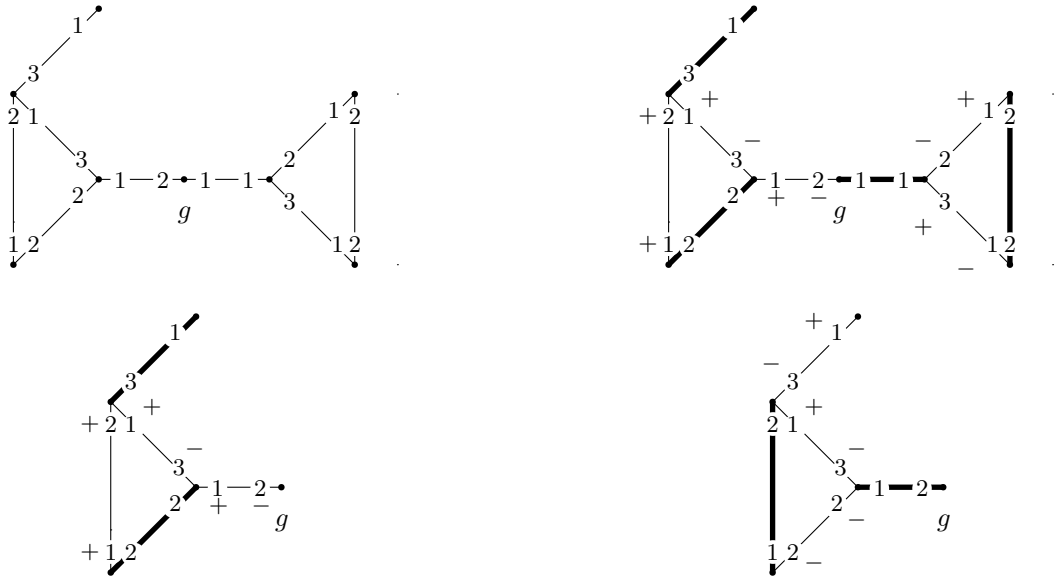

Figure 4.5: The instances and matchings from Example 2.

Consider first the matching M' *in* G' *from Figure 2, bottom left. As we already discussed,* M' *is popular in* $(G', <)$*, as it does not contain any of the forbidden structures from Theorem 13. However, it contains exactly one path* P *of* $G'_{M'}$ *that ends in* g *and has the following properties:*

- *no* $(+, +)$ *edge;*
- *no exposed node;*

- *g is matched.*

It turns out that we cannot extend M' to a popular matching of G because g being matched in M' forces P to become an alternating path in $G_{M'}$ from an exposed node to a $(+,+)$ edge – one of the forbidden structures from Theorem 13.

Now consider matching $\overline{M}$ from Figure 2, bottom left. It contains exactly one path ending in g in $G_{\overline{M}}$ with the following properties:

- *it contains a $(+,+)$ edge;*
- *it contains an exposed node g.*

Hence, following Theorem 13, we deduce that $\overline{M}$ is not popular in G'. However, even though g is not matched in this subgraph, it may be matched in G, hence there may be a popular matching of G that contains $\overline{M}$ but does not contain a forbidden structure. In fact such a matching exists – this is matching M given in Figure 2, top right.

From this example, we deduce some useful general observations.

(i) Suppose we decompose the input graph G into graphs G', G'', sharing a set S of vertices (in Example 2, $S = \{g\}$). A matching on G' cannot be extended to a popular matching if it contains a forbidden cycle from Theorem 13, or a forbidden path from Theorem 13 that does not end in a node of S.

(ii) Now consider matchings of G' that does not contain any of the structure from (i). In order to understand if / how they can be extended to a popular matching of G, we only need to remember if they contains alternating paths that end at a vertex of S, which of those paths contain a $(+,+)$ edge or an exposed node, and which edges from S are matched. Hence, we can partition matchings of G' in classes according to which paths with the features above they contain. Note that if $|S|$ is "small", the number of classes will also be "small".

(iii) We can now extend matchings from (ii) by "pairing" them with matchings from G''. In order to find a matching of maximum weight, we can keep just one representative from each of the classes defined in (ii) – the one of maximum weight. ◇

In the general case, the graph will be decomposed multiple times, following the treewidth decomposition. We next formalize and extend the ideas seen above to deal with this more complex setting.

4.6.2 Treewidth: basic definitions and facts

Let $G = (V, E)$ be a graph. A *tree decomposition* of G is a pair $(T, \mathcal{B})$ where T is a tree and $\mathcal{B} = \{B(i) : i \in V(T)\}$ is a family of subsets of V, called *bags*, one for each vertex of T, satisfying the following:

1. For each $(u, v) \in E$, there is at least one bag $B \in \mathcal{B}$ such that $(u, v) \in B$.
2. If $i \neq j \neq k \in V(T)$ are such that k is on the unique path from i to j in T then $B(i) \cap B(j) \subseteq B(k)$.

Note that the second property implies that, for any vertex $u \in V$, the bags which contain u form a subtree of T. We will sometimes abuse notation and denote by B both a vertex of $V(T)$ and the bag – i.e., the subset of V – corresponding to it.

The *width* of a tree decomposition $(T, \mathcal{B})$ is $\max\{|B| - 1 : B \in \mathcal{B}\}$. The *treewidth* of G is the minimum integer ω such that there is a tree decomposition of G of width ω.

Let $G = (V, E)$ be a graph and $(T, \mathcal{B})$ be a tree decomposition of G of width ω. Wlog we can assume that, for each pair of bags B, B' adjacent in T, $B \cap B'$ is a vertex separator of G. Following [42], we also assume that our tree decomposition is *nice*, i.e., satisfy the following properties. Edges of T are oriented as to form a *rooted tree*, i.e., we identify a special bag of the tree as being the root, and orient all edges of T away from it. For every bag B of T other than the root, we let $S(B)$ be the unique bag such that $(S(B), B)$ is an arc of the rooted tree. $S(B)$ is call the *predecessor* of B. When B is the root, we let $S(B) = \emptyset$. For each bag B' of T with (B, B') an arc

of T, we call B' a *successor* of T. Each bag has at most two successors. We will use the following result from [42], in the version given in [3].

Theorem 15. *Let $G(V, E)$ be a graph. Assume there exists a tree decomposition of G of width ω with t bags. Then in time $O(\omega^2(|V| + t))$ we can find a nice tree decomposition of width ω with at most $O(\omega|V|)$ nodes.*

Thus, from now on, we will always assume that our tree decomposition is nice (hence rooted). Moreover, since the running time of producing a nice decomposition is dominated by the running time of the algorithm, we will ignore the former.

Let T' be a subtree of T, and $V(T')$ the set of nodes contained in at least a bag of T'. We say that T' is a *closed subtree* of T if, whenever B belongs to T' and B' is a successor of B, then B' also belongs to T'. By connectivity, for each node B of a closed subtree T', $S(B)$ also belongs to T', with the exception of at most one node that we call the *head* of T' and denoted by $H(T')$. If $T' \neq T$, the predecessor of $H(T')$ exists and it is also called the *predecessor of* T' and denoted by $S(T')$. Note that each bag B is the head of exactly one closed subtree of T, that we denote by T_B.

4.6.3 Locally popular matchings, configurations, and tipping points

Fix an instance $(G, <)$ of the roommate problem with $G = (V, E)$, and a weight vector w over the edges. By standard perturbation techniques, we can assume without loss of generality that $w(F) \neq w(F')$ for each $F \neq F' \subseteq E$.

We say that M is a *U-matching* if for all edges $(u, v) \in M$, we have $\{u, v\} \cap U \neq \emptyset$. The *$U$-matching induced by M* is the set $M_U := \{e \in M : e \cap U \neq \emptyset\}$.

Let S be a separator of G, and X so that $G[X]$ is a connected component of $G \setminus S$. Consider a $(X \cup S)$-matching M of G, with labels attached to each $e \in E(G) \setminus M$ as described in Section 4.3. We say that M is *(S, X)-locally popular* if none of the structures (i), (ii), and (iii) from Theorem 13 is a subgraph of $G_M[X \cup S]$. We remark that, in the graph $G_M[X \cup S]$, for a node $v \in S$ matched in M to a neighbor outside $X \cup S$, the labels of edges of $G_M[X \cup S]$ incident to v are a function of the edge $(v, M(v))$ (however note that the edge $(v, M(v))$ is not in $G[X \cup S]$). In particular, a

node v that is not matched in $G_M[X \cup S]$ may still be M-covered (hence not M-exposed). If this happens, v cannot be the M-exposed node in the path (iii) from Theorem 13, and some or all of the edges from $G[X \cup S]$ incident to v may have a "$-$" label.

Example 2, continued. In Figure 2, let $S = \{g\}$, X all the nodes to the left of g, M be the matching from the top right and $\overline{M} = M_{(X \cup S)}$. g is $\overline{M}$-covered. ◇

The following easy lemma shows that a certain monotonicity property (absent in popular matchings) is recovered by (S, X)-locally popular matchings, and that popular matchings arise as a special case of locally popular. In particular, if M is a popular matching, for S, X as above we have that $M_{X \cup S}$ is (S, X)-locally popular.

Lemma 4.6.1. *Let S (resp. S') be a separator of a graph $G(V, E)$, and $X \subseteq V$ (resp. $X' \subseteq V$) so that $G[X]$ (resp. $G[X']$) is a connected component of $G \setminus S$ (resp. $G \setminus S'$). Assume that $X' \cup S' \supseteq X \cup S$. Then the following holds.*

(i) Let M be a matching in G. Then M is $(\emptyset, V)$-locally popular if and only if it is popular.

(ii) Let M' be a (S', X')-locally popular matching. Then $M'_{X \cup S}$ is an (S, X)-locally popular matching.

Proof. (i). It follows by definition and from the fact that $G_M[\emptyset \cup V] = G_M$.

(ii). Suppose M is not (S, X)-locally popular. Then, one of the structures from Theorem 13 is a subgraph of $G_M[X \cup S]$ — call it P. We claim that P is a forbidden structure in $G_{M'}[X' \cup S']$ (again, in the sense of Theorem 13), hence M' is not (S', X')-locally popular, a contradiction.

Indeed, $X \cup S \subseteq X' \cup S'$, and none of the edges of $M' \setminus M$ is incident to $X \cup S$ by definition. We deduce $G_M[X \cup S] = G_{M'}[X \cup S]$ and a node of $X \cup S$ is M-exposed if and only if it is M'-exposed. This concludes the proof. □

We now introduce the concepts of *configuration* and *tipping point* in order to partition the family of (S, X)-locally popular matchings into a (small) number of classes. These definitions are related to the existence of certain structures that are not forbidden in a locally popular matching,

but restrict the possibility of them to popular matchings in the whole graph. We then illustrate these definitions on the instance from Example 2.

Again, since all graphs we deal with are simple, we represent paths and cycles as ordered set of nodes, with the first node of a cycle coinciding with the last. A path is *trivial* if it is composed of a single node; *non-trivial* otherwise. This representation also allows us to distinguish between the *first* and *last* node of a path. We will say that *e is an edge of P* if it is an edge between two consecutive nodes of P. Let $U \subseteq V$. Two paths $P = (v_1, \dots, v_k)$, $P' = (v'_1, \dots, v'_{k'})$ are called *U-disjoint* if $P \cap P' \cap U = \emptyset$. P, P' are said to be *internally vertex-disjoint* if, for all $i \in [k]$ and $j \in [k']$, we have $v_i \neq v'_j$, with possibly the exception of $(i, j) \in \{(1, 1), (1, k'), (k, 1), (k, k')\}$.

A non-trivial path $P = (v_1, \dots, v_k)$ is a *U-path* if $\emptyset \neq P \cap U \subseteq \{v_1, v_k\}$, i.e., either the first or the last node of P (or both) belong to U, and all other vertices of P do not belong to U. Note that an edge between two nodes of U is a U-path.

From now on, we also fix M to be a $(X \cup S)$-matching of G. Let $\mathcal{P}_M$ be the set of M-alternating paths of $G_M[X \cup S]$. Recall that we identify each path with its ordered set of nodes, or equivalently its ordered set of edges. We associate to each $P \in \mathcal{P}_M$ a 2-dimensional *parity* vector π, where the first component of π is defined to be 0 if the first edge of P is an edge of M, 1 otherwise. Similarly, the second component of the parity vector π takes values 1 if the last edge of the path is an edge of M, and 0 otherwise. We also associate to P a 2-dimensional *level* vector $\ell = (e, p)$, where $e \in \{0, 1, 2\}$ is the number of M-exposed nodes of P, and $p \in \{0, 1, \infty\}$ denotes the number of $(+, +)$ edges of P, where $p = \infty$ if it contains 2 or more such edges. Note that not all parity-level combinations are possible. Moreover, if $p = \infty$, then M is not (S, X)-locally popular. For some $k \in \mathbb{N}$, let

$$U = ((u_1, v_1), (u_2, v_2), \dots, (u_k, v_k)), \quad L = (\ell_1, \dots, \ell_k), \quad \Pi = (\pi_1, \pi_2, \dots, \pi_k) \tag{4.1}$$

be as follows:

- $u_1, v_1, u_2, \dots, v_k \in S \cup \{\emptyset\}$, under the additional condition that $u_i \neq v_i$ for all $i \in [k]$, all

pairs are different, and each node of S can appear at most twice in the collection.

- $L = (\ell_1, \ldots, \ell_k)$, where, for $i \in [k]$, $\ell_i \in \{0, 1, 2\} \times \{0, 1\}$.
- $\Pi = (\pi_1, \pi_2, \ldots, \pi_k)$, where, for $i \in [k]$ $\pi_i \in \{0, 1\} \times \{0, 1\}$.

The triple $C = (U, L, \Pi)$ is called an (S, X)-*configuration*.

We say that M is *active at* C if there exist pairwise X-disjoint S-paths $P^1, \ldots, P^k \in \mathcal{P}_M$, such that, for $i \in [k]$, P^i:

- is of level ℓ_i and parity π_i;
- starts at $u_i \in S$ if $u_i \neq \emptyset$ and at some node of X otherwise; ends at $v_i \in S$ if $v_i \neq \emptyset$ and at some node of X otherwise.

We call those paths the *certificate*[2] *of* C *at* M.

Let $\{C_1, \ldots, C_q\}$ be the collection of all (S, X)-configurations at which M is active. We call

$$(\{C_1, \ldots, C_q\}, M_S)$$

the (S, X)-*tipping point of* M.

Example 2, continued. In Figure 2, let $S = \{g\}$, X all the nodes to the left of g, M be the matching from the top right and $\overline{M} = M_{(X \cup S)}$. Let G' be the graph in the bottom left. We want to describe the $(\{g\}, X)$-tipping point of matching $\overline{M}$. We list all the configurations at which $\overline{M}$ is active. There is an alternating path P in $G'_{\overline{M}}$ composed of the single edge $e \in \overline{M} \cap E(G')$ incident to g, starting from a matched node. Hence, if we let $C_1 := (\underbrace{((\emptyset, g))}_{U}, \underbrace{((1, 1))}_{L}, \underbrace{(0, 0)}_{\Pi})$, we can conclude that $\overline{M}$ is active at C_1. We can extend P to an alternating path in $G'_{\overline{M}}$ with two, three, or four edges, including another $(+, +)$ edge in the latter two cases. $\overline{M}$ is therefore also active at $C_2 :=$

[2]Note that, if M is active at C, it is also active at the configurations, obtained, e.g., by permuting entries of U (and of L and Π accordingly). This fact (and others) causes some redundancy. As this redundancy does not affect our analysis, we do not eliminate it.

$(((\emptyset, g)), ((1,1)), (1,0))$, $C_3 := (((\emptyset, g)), ((2,1)), (0,0))$, and $C_4 := (((\emptyset, g)), ((2,1)), (1,0))$. Hence, the $(\{g\}, X)$-tipping point of $\overline{M}$ is $(\{C_1, \ldots, C_4\}, \emptyset)$. $\diamond$

Next lemma gives some basic properties of configurations and tipping points.

Lemma 4.6.2. *Let G, X, S be as in Lemma 4.6.1.*

(i) Let M be an (S, X)-locally popular matching. The (S, X)-tipping point of M is uniquely defined.

(ii) There exists a function $g : \mathbb{N} \to \mathbb{N}$ such that there are $g(|S|)$ triples (U, L, Π) with the property that each (S, X)-configuration at which an (S, X)-matching is active is one of those triples.

(iii) The collection of (S, X)-tipping points of M, where M ranges over all (S, X)-locally popular matchings, has size at most $g(|S|)|V|^{|S|}$, where g is the function from part (ii).

Proof. (i) follows by definition. To show (ii), observe that the number of (S, X)-configurations (U, L, Π) is a function of the size of S only, since all pairs from U are different, and each node of S can appear in at most two pairs from U. To show (iii), observe that the number of S-matchings of G is upper bounded by $|V|^{|S|}$. $\square$

Now fix G, S, X, M as above, and let $\mathcal{T}$ be the (S, X)-tipping point of M. M is said to be the *$\mathcal{T}$-leader* if it is the matching of maximum weight among all (S, X)-locally popular matchings whose (S, X)-tipping point is $\mathcal{T}$. Due to the initial perturbation of costs, note that there is at most one $\mathcal{T}$-leader for each tipping point $\mathcal{T}$.

The following crucial lemma implies that, in order to find a min cost popular matching, it suffices to consider matchings that induce $(X \cup S)$-matchings that are $\mathcal{T}$-leaders, for some (S, X)-tipping point $\mathcal{T}$. Its proof is given in Section 4.6.6.

Lemma 4.6.3. *Let G, X, S, M be as in Lemma 4.6.2, and $\mathcal{T}$ the (S, X)-tipping point of M. Let S' be a vertex separator of G, and $X' \subseteq V$ be so that $G[X']$ is a connected component of $G \setminus S'$, with the property that $X' \supseteq X$ and $X' \cup S' \supseteq X \cup S$ (possibly $S' = \emptyset$ and $X' = V$). Let M' be a*

(S', X')-locally popular matching so that $M'_{X \cup S} = M$, and let $\mathcal{T}'$ be the (S', X')-tipping point of M'. If M' is the $\mathcal{T}'$-leader, then M is the $\mathcal{T}$-leader.

4.6.4 The algorithm

We now give our algorithm for computing a maximum weight popular matching, see Algorithm 8. Note that Algorithm 8 relies on the subroutine `Update` described in Algorithm 9, and the implementations of some other subroutines are not completely defined in the algorithm itself. We give a formal description of those together with a complexity analysis in the proof of Theorem 16.

Algorithm 8 takes as input a graph G with strict preference lists as usual, and a nice tree decomposition $(T, \mathcal{B})$ of G. It iteratively constructs the sequence of closed subtrees T_B of T, with the first T_B corresponding to a leaf B of T, and the last to the root.

For each T_B, let $S = B \cap S(B)$ and $X = V(T_B) \setminus S$. (recall that V_B is the unique closed subtree of T whose head is B). Steps 4 – 13 construct the set $\mathcal{L}_B$ containing, for each (S, X)-tipping point $\mathcal{T}$, the pair $(M, \mathcal{T})$, where M is the $\mathcal{T}$-leader. $\mathcal{L}_B$ is constructed via multiple calls to the subroutine `Update`, that guarantees that all pairs $(M, \mathcal{T})$ in $\mathcal{L}_B$ are, at each step, the (S, X)-locally popular matching with tipping point $\mathcal{T}$ of maximum weight among those enumerated so far. If we establish that $\mathcal{L}_B = \emptyset$, the algorithm halts and concludes that G has no popular matching. Finally, of all matchings M such that $(M, \mathcal{T}) \in \mathcal{L}_B$ – with B being the root – the one of maximum weight is output.

With a little abuse of notation, we will write $M \in \mathcal{L}_B$ if $(M, \mathcal{T}) \in \mathcal{L}_B$ for some $\mathcal{T}$. Similarly, we write $\mathcal{L}_B = \mathcal{L}_B \cup \{M\}$ to mean that $(M, \mathcal{T})$ is added to $\mathcal{L}_B$ for an appropriate $\mathcal{T}$ (that is clear from the context), and similarly for $\mathcal{L}_B \setminus \{M\}$.

4.6.5 Analysis

The goal of this section is to prove the following result, which together with Theorem 15 implies Theorem 12.

Algorithm 8

Require: A graph G, together with, for each node $v \in V$, a strict ranking of the neighbors of v, and a function $w : E \to \mathbb{Z}$. A nice tree decomposition $(T, \mathcal{B})$ of G.

Ensure: A popular matching of maximum weight in G.

For all $B \in \mathcal{B}$, label B as *unflagged*.
Choose an unflagged bag B with no unflagged successors, and flag B.
Let $S = B \cap S(B)$, $X = V(T_B) \setminus S$.
if B has no successor in T **then**
  for all B-matchings M^* of G **do**
    `Update`$(M^*, \mathcal{L}_B)$
  end for
else
  Let $S_1 = B \cap B_1$ and (possibly) $S_2 = B \cap B_2$, where B_1 and (possibly) B_2 are the successors of B.
  for all B-matchings M of G, all $M_1 \in \mathcal{L}_{B_1}$ and (possibly) $M_2 \in \mathcal{L}_{B_2}$ **do**
    if $(M_1)_{S_1} = M_{S_1}$ and (possibly) $(M_2)_{S_2} = M_{S_2}$ **then**
      Let $M^* = M \cup M_1 \cup M_2$.
      `Update`$(M^*, \mathcal{L}_B)$
    end if
  end for
end if
if $\mathcal{L}_B = \emptyset$ **then**
  output: G has no popular matching.
end if
if there is a bag $B \in \mathcal{B}$ that is unflagged **then**
  Go to Step 2.
end if
Let B be the head of T. Output the matching of maximum weight from $\mathcal{L}_B$.

Theorem 16. *Algorithm 8 is correct. If the treewidth of $G = (V, E)$ is upper bounded by a constant ω, then it can be implemented to run in time $|V|^{O(\omega)} \cdot g(\omega+1)^{O(1)}$, where g is the function from Lemma 4.6.2.*

We assume throughout the proof that every bag B that is not a leaf has exactly two successors, as the (simpler) case where some B has only one successor follows in a similar fashion (recall that we assume that the decomposition is nice, hence no more than two successor per bag are possible). In the following, we will often fix a bag B and write $S := B \cap S(B)$ and $X := V(T_B) \setminus S$; if B is not a leaf, we denote by B_1 and B_2 its successors, and write $S_i := B \cap B_i$, $X_i = V(T_{B_i}) \setminus S_i$ for $i = 1, 2$.

Well-definedness. Since each bag is flagged at most once, the algorithm terminates. In order to

Algorithm 9 `Update`

Require: M^*, $\mathcal{L}_B$

if M^* is an (S, X)-locally popular matching **then**
  Find the (S, X)-tipping point $\mathcal{T}$ of M^*.
  if there exists $M' \in \mathcal{L}_B$ whose tipping point is $\mathcal{T}$ **then**
    if $w(M^*) > w(M')$ **then**
      Set $\mathcal{L}_B = \mathcal{L}_B \setminus \{M'\} \cup \{M^*\}$.
    end if
  else
    Set $\mathcal{L}_B = \mathcal{L}_B \cup \{M^*\}$.
  end if
end if

show that the algorithm is well-defined, it suffices therefore to show that X, S as defined in Step 3 are such that S is a cutset of G and X a connected component of $G \setminus S$ This follows from the fact that we assumed that, for each bag B, $S(B) \cap B$ is a clique cutset of G, see Section 4.6.2.

Correctness. We show that, at the end of the iteration where bag B is flagged,

($*$) $\mathcal{L}_B$ contains exactly all $\mathcal{T}$-leaders, for all (S, X)-tipping points $\mathcal{T}$ for which a $\mathcal{T}$-leader exists.

Once ($*$) is proved, we can apply it when B is the root to conclude that $\mathcal{L}_B$ contains only popular matchings (by Lemma 4.6.1), and the one of maximum weight among those is the popular matching of maximum weight. Again by Lemma 4.6.1, if $\mathcal{L}_B = \emptyset$ at the end of the iteration where B is flagged, then G has no popular matching, and the output of Algorithm 8 is again correct.

The proof of ($*$) is by induction on the number of nodes n_B of T_B. If $n_B = 1$, then the condition from the **if** statement in Step 4 is verified. In this case, the statement is immediate, since we enumerate over all possible B-matchings of G, check those that are (S, X)-locally popular, and for each (S, X)-tipping point $\mathcal{T}$, keep the (S, X)-locally popular matching of maximum weight active at $\mathcal{T}$.

Now suppose $n_B > 1$. By induction hypothesis, for $i = 1, 2$, all matchings that are stored in $\mathcal{L}_{B_i}$ are exactly all $\mathcal{T}_i$-leaders, for all (S_i, X_i)-tipping points $\mathcal{T}_i$ such that there is at least a (S_i, X_i)-locally popular matching active at $\mathcal{T}_i$. By construction, if $(M, \mathcal{T}) \in \mathcal{L}_B$, then M is an (S, X)-locally

popular matching active at $\mathcal{T}$. Now let $\mathcal{T}$ be an (S, X)-tipping point for which a $\mathcal{T}$-leader $\widehat{M}$ exists. It suffices to show that one of the matchings M^* constructed at Step 12 is $\widehat{M}$.

Let $i \in \{1, 2\}$. Since $\widehat{M}$ is (S, X)-locally popular and $X_i \cup S_i \subseteq X \cup S$, matching $M_i := \widehat{M}_{X_i \cup S_i}$ is also (S_i, X_i)-locally popular by Lemma 4.6.1. By Lemma 4.6.3, M_i is a $\mathcal{T}_i$-leader for an appropriate tipping point $\mathcal{T}_i$. By induction, $M_i \in \mathcal{L}_{B_i}$. On the other hand, $M := \widehat{M}_B$ is a B-matching of G such that $M_{S_i} = (M_i)_{S_i}$. Since we enumerate all triples given by a B-matching of G, a matching from $\mathcal{L}_{B_1}$, and a matching from $\mathcal{L}_{B_2}$, matching $\widehat{M} = M \cup M_1 \cup M_2$ is eventually constructed in Step 12.

Running time. We now bound the running time of Algorithm 8. Before investigating the complexity of each step, we state and prove a fact that we will use extensively: if we are given an $(X \cup S)$-matching M in a graph H and $|X \cup S|$ is bounded, then we can check efficiently if M is (S, X)-locally popular. Moreover, if M is (S, X)-locally popular, then we can efficiently find the (S, X)-tipping point at which M is active.

Claim 4.6.4. *Let $H = (U, F)$ be a graph, S a vertex separator of H, X a connected component of $H \setminus S$. Let M be a $(X \cup S)$-matching of H. Let $|X \cup S| \leq k$. Let g be defined as in Lemma 4.6.2. Then in time $O(g(k) \cdot 2^{O(k^2)}) + O(|F|)$, we can:*

(i) check if M is (S, X)-locally popular in H and, if it is,

(ii) find the (S, X)-tipping point (in H) at which M is active.

Proof of claim. Building graph $H_M[X \cup S]$ takes $O(k^2) + O(|F|)$ time. In time $O(2^k)$ we can enumerate all paths and cycles of $H_M[X \cup S]$, and for each such subgraph, it takes time $O(k)$ to decide if it is one of the forbidden substructures that show that M is not (S, X)-locally popular. This proves (i). Suppose now that we concluded that M is (S, X)-locally popular. Let C be an (S, X)-configuration. By definition, C is composed of $O(k)$ pairs. In order to decide if M is active at C, we can therefore proceed as follows: select a family $\mathcal{P}$ of $O(k)$ paths in H_M; check in time $O(k^2)$ if $\mathcal{P}$ is a certificate for M at C. Since the number of paths in $H_M[X \cup S]$ is $O(2^k)$ as discussed above, in time $2^{O(k^2)}$ we can decide if M is active at C. By Lemma 4.6.2, we can pick

all (S, X)-configurations from a candidate set of size $g(k)$. Moreover, M_S can be computed in time $O(|F|)$. (ii) follows. ◇

We now discuss the complexity of Algorithm 8. At each repetition of Steps 2–18 we flag one node of T. Since no node of T is ever unflagged after Step 1, and T has $O(V)$ nodes, it suffices to find the bottleneck in steps 2–18.

Note that each repetition of Steps 2–18 – that we call *iteration* – is associated to a node B of T. Hence fix one such node, and recall that $|B| \leq \omega + 1$. Steps 2–3 require simple manipulations of nodes of T, hence we do not investigate their complexity further.

Suppose first that B is a leaf of T, i.e., it satisfies the **if** condition of Step 4. This implies that $|X \cup S| = |B| = |V(T_B)| \leq \omega + 1$. Enumerating all B-matchings takes time $O(|V|^{\omega+1})$. We therefore assume we have one such matching M^* and bound the complexity of the `Update` subroutine. Because of Claim 4.6.4, Step 1 and Step 2 can be performed in time $g(\omega + 1) \cdot 2^{O(\omega^2)}$. Now observe that, for each (S, X)-tipping point $\mathcal{T}$, throughout the algorithm, we have at most one entry $(M, \mathcal{T}) \in \mathcal{L}_B$. Hence, by Lemma 4.6.2, $|\mathcal{L}_B| \leq g(|B|)|V|^{|B|} \leq g(\omega + 1)|V|^{\omega+1}$. Step 3 of the `Update` subroutine can therefore be performed in time $|\mathcal{L}_B| \cdot \log(|\mathcal{L}_B|)$ for an appropriate encoding of the (S, X)-tipping points. If we find a matching M' whose tipping point coincide with the tipping point of M^*, then their costs can be computed and compared in time $O(|E|)$. We conclude that, when B is a leaf, the associated iteration can be performed in time at most $g(\omega + 1)^{O(1)} \cdot |V|^{O(\omega)}$.

Now assume that B is not a leaf of T, i.e., we execute Steps 8–13. The number of B-matchings is $|V|^{O(\omega)}$. Because of the correctness of the algorithm, for $i = 1, 2$, $|\mathcal{L}_{B_i}|$ is upper bounded by the number of (S_i, X_i)-tipping points (where $X_i = V(T_{B_i})$), which by Lemma 4.6.2 are $g(|S_i|)|V|^{|S_i|} \leq g(\omega + 1)|V|^{\omega+1}$. The condition from Step 11 can be verified in time $O(|E|)$. Hence, repeating the arguments for the case when B is a leaf, we conclude that the running time of Steps 8–13 is upper bounded by $g(\omega + 1)^{O(1)} \cdot |V|^{O(\omega)}$ times the time needed to solve the following problem.

Problem 6. Given: *An (S, X)-configuration C and a matching $M^* = M \cup M_1 \cup M_2$, where M is a B-matching M of G and $M_i \in \mathcal{L}_{B_i}$ with $(M_i)_{S_i} = M_{S_i}$ for $i = 1, 2$.* ***Decide****: if M^* is (S, X)-locally*

popular and if it is active at C.

Note that we cannot solve Problem 6 by directly employing Lemma 4.6.4, since $|X \cup S|$ may be linear in $|V|$. Instead, we show how to reduce Problem 6 to an analogous problem on a graph whose size we can control.

Fix an (S_1, X_1)-configuration $C_1 = (U_1, L_1, \Pi_1)$ in G at which M_1 is active, and an (S_2, X_2)-configuration $C_2 = (U_2, L_2, \Pi_2)$ in G at which M_2 is active. Consider the graph H and matching $M(H)$ obtained as follows. Start from $H = G_M[B]$ and the corresponding matching $M(H) = M_{B\setminus(S_1\cup S_2)}$, and let (u, v) be the first pair from U_1. Assume $u, v \neq \emptyset$ are matched in $M(H)$, $\ell_1 = (0, 1)$, $\pi_1 = (0, 0)$. By the definition, there is an S_1-path P in $G[X_1 \cup S_1]$ that: starts at u and ends at v (since $U_1 = (u, v)$) with edges of $E \setminus M$ (since $\pi_1 = (0, 0)$); contains no unmatched nodes and one $(+, +)$ edge (since $\ell_1 = (0, 1)$). Note that $P \cap B = \{u, v\}$. Add to H and $M(H)$ new nodes u', w', z', v', matching edges (u', w'), (z', v'), $(+, +)$ edge (w', z'), and $(+, -)$ edges (u, u') and (v, v'). Note that (u, u', w', z', v', v) is a $M(H)$-alternating S_1-path of parity π_1 and level ℓ_1 starting at u and ending at v. We call it the *shortcut* of the path P. Repeat this operation for all pairs from U_1 and U_2, adding at each step new nodes to H (the other cases following in a similar fashion to the one described above). Note that these operations adds to H $O(|B|) = O(\omega)$ nodes, since all endpoints of paths that are active at a configuration belong to $S_1 \cup S_2 \subseteq B$, and each node can only be the endpoint of at most two paths that are active. Hence, the graph H and matching $M(H)$ can be constructed in time $O(|E| + \omega)$. The following two claims imply that we can decide if M is (S, X)-locally popular and, if it is, compute its tipping point by investigating matching $M(H)$ on graph H.

Claim 4.6.5. *M^* is not an (S, X)-locally popular matching in G if and only if there exist an (S_i, X_i)-configuration C_i at which M_i is active for $i = 1, 2$, such that, if we construct graph H and matching $M(H)$ as above, then $M(H)$ is not $(S, V(H) \setminus S)$-locally popular in H.*

Claim 4.6.6. *Let M^* be (S, X)-locally popular in G. Then M^* is active at C if and only if there exist a (S_i, X_i)-configuration C_i at which M_i is active for $i = 1, 2$ such that if we construct graph H*

and matching $M(H)$ as above, then $M(H)$ is active[3] *at C.*

The proof of Claim 4.6.5 is given in Appendix B.3.2. The proof of Claim 4.6.6 is analogous to the proof of Claim 4.6.5 and is therefore omitted.

Using Claim 4.6.5 and Claim 4.6.6, we solve Problem 6 as follows. By Claim 4.6.5, we can check if M^* is an (S, X)-locally popular matching in G by checking, for each pair of (S_i, X_i)-configurations C_i at which M_i is active for $i = 1, 2$, if the corresponding $M(H)$ is $(S, V(H) \setminus S)$-locally popular in H. For $i = 1, 2$, the number of (S_i, X_i)-configurations is at most $g(\omega + 1)$. Since the number of nodes of H is $O(\omega)$, testing if $M(H)$ is $(S, V(H) \setminus S)$-locally popular in H can be done in time $|V|^{O(1)} + g(\omega+1)^{O(1)}$ by Claim 4.6.4. Similarly, if M^* is (S, X)-locally popular in G, we can find its tipping point in time $|V|^{O(1)} + g(\omega+1)^{O(1)}$ by repeatedly applying Claims 4.6.6 and Claim 4.6.4. Hence, Problem 6 can be solved in time $(|V|^{O(1)}+g(\omega+1)^{O(1)}$, thus the iteration when B is flagged can be performed in time $|V|^{O(\omega)} \cdot g(\omega + 1)^{O(1)}$, concluding the proof of Theorem 16.

4.6.6 Proof of Lemma 4.6.3

We prove Lemma 4.6.3 by showing the contrapositive: we assume that M is not the $\mathcal{T}$-leader, and show that M' is not the $\mathcal{T}'$-leader. Let N be the (S, X)-locally popular matching that is the $\mathcal{T}$-leader.

Claim 4.6.7. *No edge from $M' \setminus M$ is incident to nodes of $X \cup S$, while all edges from N are incident to nodes of $X \cup S$.*

Proof of claim. $(M' \setminus M) \cap \{\delta(v) : v \in X \cup S\} = (M' \setminus M'_{X \cup S}) \cap \{\delta(v) : v \in X \cup S\} = \emptyset$, while $N \subseteq \{\delta(v) : v \in X \cup S\}$ by definition. ◇

Define $N' := N \cup (M' \setminus M)$.

Claim 4.6.8. *N' is an $(X' \cup S')$-matching of G and $N'_{X \cup S} = N$.*

[3] Here we interpret C as an $(S, V(H) \setminus S)$-configuration in H, which is allowed since $S \subseteq V(H)$.

Proof of claim. If follows from Claim 4.6.7 that the edges of N' incident to nodes of $X \cup S$ are all and only those from N. Hence, N' is a matching and $N'_{X \cup S} = N$.

By hypothesis, $X \cup S \subseteq X' \cup S'$. Hence, N has no edge in $G \setminus (X' \cup S')$. Moreover, M' is by hypothesis an (S', X')-locally popular matching, so it is an $(X' \cup S')$-matching. Hence, N' is an $(X' \cup S')$-matching. $\diamond$

We next show that N' is a (S', X')-locally popular matching whose $(X' \cup S')$-tipping point is $\mathcal{T}'$. We then have:

$$w(N') = w(N) + w(N' \setminus N) > w(M) + w(M' \setminus M) = w(M'),$$

concluding the proof.

The first two of the next claims immediately follow by construction, Claim 4.6.7, and Claim 4.6.8.

Claim 4.6.9. *$G_{M'}[X \cup S] = G_M[X \cup S]$ and $G_{N'}[X \cup S] = G_N[X \cup S]$.*

Claim 4.6.10. *$M' = M \cup (N' \setminus N)$, and the latter is a disjoint union.*

Claim 4.6.11. *Let $e \in E$. Assume first that $e \in \delta(v)$ for some $v \in S$. Then:*

(i) $e \in M$ if and only if $e \in N$.

(ii) Suppose $e \notin M$ and let $\star \in \{+, -\}$ be the label of e at v wrt M. Then $e \notin N$ and $\star$ is also the label of e at v wrt N.

Suppose now instead that e is not incident to a node of X. then

(iii) $e \in N'$ if and only if $e \in M'$.

(iv) Let $e \notin N'$. Then $e \in E(G_{N'})$ if and only if $e \in E(G_{M'})$. Moreover, if $e \in E(G_{N'})$, then it has the same labels in $G_{N'}$ and in $G_{M'}$.

Proof of claim. (i)-(ii). immediately follow from $M_S = N_S$, which holds by definition of tipping point.

(iii). Suppose first $e \in N$. Since e is not incident to X and N is an $(X \cup S)-$matching, we have $e \in N_S = M_S \subseteq M \subseteq M'$, where equality holds by hypothesis and inclusions by definition. Else, $e \in (N' \setminus N) \subseteq M'$ by Claim 4.6.10. This shows the "only if" direction. Switching the roles of M and N shows the "if" direction.

(iv). Let $e = uv$. By part (iii), $e \notin M'$. Suppose first $u \notin S$. Then no edge incident to u is also incident to X. Hence, $N'(u) = M'(u)$ by applying part (iii) to edges incident to u. We deduce that the label of e at u coincides in M' and N'. Suppose now that $u \in S$. Since $e \notin N', M'$, we have that $e \notin N, M$. The label of e at u in G_M and G_N coincide by part (ii). ◇

Our first step to prove that N' and M' have the same tipping point is the next proposition, that shows how to obtain certain N'-alternating paths from analogous M'-alternating paths. The proof of next proposition is given in Appendix B.3.1

Proposition 4.6.12. *Let $k \in \mathbb{N}$ and $P^1, \dots, P^k \in \mathcal{P}_{M'}$ be pairwise internally vertex-disjoint and X-disjoint M'-alternating paths in $G_{M'}[X' \cup S']$. Assume that, for $j \in [k]$, path P^j is of level $\ell_j \in \{0,1,2\} \times \{0,1\}$ and parity $\pi_j \in \{0,1\} \times \{0,1\}$, that every node appears at most twice in $P^1 \dots, P^k$, no pair of paths have exactly the same endpoints, and no path is contained in X. Then there exist pairwise internally vertex-disjoint and X-disjoint paths $Q^1, \dots, Q^k \in \mathcal{P}_{N'}$ with the following properties:*

(i) For $j \in [k]$, Q^j is of level ℓ_j and parity π_j, and $P^j \cap S' = Q^j \cap S'$.

(ii) For $j \in [k]$, let u_j (resp. v_j) be the first (resp. last) node of P^j. If u_j (resp. v_j) $\notin X$, then u_j (resp. v_j) is the first (resp. last) node of Q^j. If u_j (resp. v_j) $\in X$, the first (resp. last) node of Q^j also belongs to X.

Moreover,

(iii) (i)-(ii) also hold if we switch the roles of M' and N'.

We now have all the tools to prove that N' is a (S', X')-locally popular matching whose tipping point is $\mathcal{T}'$, concluding the proof of the lemma.

(S', X')-local popularity. Let us first show that N' is (S', X')-locally popular. Suppose it is not, then there exists a forbidden path or cycle P as in item (i), (ii), or (iii) from Theorem 13 that is contained in $G_{N'}[X' \cup S']$. Since (by Claim 4.6.9) $G_N[X \cup S] = G_{N'}[X \cup S]$ and $v \in X \cup S$ is N-exposed if and only if it is N'-exposed, we deduce $P \not\subseteq X \cup S$, else N would not be (S, X)-locally popular, a contradiction.

Let us now also observe that $P \cap X \neq \emptyset$. In fact, suppose $P \cap X = \emptyset$. Then Claim 4.6.11, part (iii) and (iv) imply that M' is not (S', X')-locally popular, a contradiction.

Suppose first P is a path: take one that is inclusionwise minimal among all paths that certify that N' is not (S', X')-locally popular. We can then write P as the concatenation of (P^1, P^2) of levels $\ell_1, \ell_2 \neq \infty$, so that the first node of P^2 (which coincides with the last node of P^1) does not belong to X. By what discussed above, we can assume that both P_1 and P_2 exist and are non-trivial (possibly, by reversing the path). P^1, P^2 satisfy the hypothesis of Proposition 4.6.12 with M' replaced by N' (which is allowed, by part (iii) of the proposition) and $k = 2$. Indeed, P^1, P^2 are by construction internally vertex-disjoint, X-disjoint M'-alternating paths from $G_{M'}[X' \cup S']$, and each of them has at most one $(+, +)$ edge since we choose P to be minimal. Let Q^1, Q^2 be the paths whose existence is guaranteed by Proposition 4.6.12. We claim that the concatenation of (Q^1, Q^2) is a forbidden path in $G_{M'}[X' \cup S']$, a contradiction to the fact that M' is (S', X')-locally popular.

First, observe that the last node of Q^1 coincides with the last node of P^1 (since the latter does not belong to X) which coincides with the first node of P^2, which coincides with the first node of Q^2 (again, since it does not belong to X). We now argue that Q^1 and Q^2 have no other node in common. We know they are internally vertex-disjoint and X-disjoint. Hence, if the first node of P^1 or the last node of P^2 belongs to X, we are done. Suppose not. Then, by Proposition 4.6.12, the first node of Q^1 is the first node of P^1, which is different by hypothesis (P is a path) from the last node of P^2, which coincides, again by Proposition 4.6.12, with the last node of Q^2. Hence, the

concatenation of (Q^1, Q^2) is a path of G, that we denote by Q. Again by Proposition 4.6.12, Q^1, $Q^2 \in \mathcal{P}_{M'}$, and the parity and level of Q^1 (resp. Q^2) coincide with the parity and level of P^1 (resp. P^2). Hence $Q \in \mathcal{P}_{M'}$ is the required forbidden path, obtaining the desired contradiction.

Now suppose that P is a cycle. The argument follows in a similar fashion, writing P as the concatenation of (P^1, P^2, P^3) with $P^1 \cap X = P^2 \cap X = \emptyset$, and the endpoints of P_3 lying in S (note that we can assume that P contains exactly one $(+,+)$ edge, else it also contains a path with two $(+,+)$ edges and we are back to the previous case).

Tipping point. Let us now show that $\mathcal{T}'$ is the tipping point of N'. First, observe that $X \subseteq X'$ implies $S' \cap X = \emptyset$. For $v \in S' \cap S$, we know that $N'(v) = N(v) = M(v) = M'(v)$, where the first equality holds by construction, the second since N and M have the same tipping point, and the third again by construction. For $v \in S' \setminus S$, we have $N'(v) = M'(v)$ by construction. Hence $N'_{S'} = M'_{S'}$. Now let $C = \{U, L, \Pi\}$ be a (S', X')-configuration at which M' is active, with U, L, Π as in (4.1). We will show that N' is also active at C. A symmetric argument shows that, if N' is active at C, then so is M', concluding the proof.

Take paths $P^1, \dots, P^k$ that are the certificate of C at M'. We claim that those paths satisfy the hypothesis of Proposition 4.6.12. They are, by definition, S'-paths that are pairwise X'-disjoint. This implies that they are pairwise internally vertex-disjoint and X-disjoint (using again $X \cap S' = \emptyset$). The X'-disjointness implies that each node from X' appears at most once. Moreover, again by construction, each node from S' appears at most twice. Hence, every node appears at most twice in $P^1, \dots, P^k$. Moreover, by definition of U, no two paths have the same endpoints.

Let $Q^1, \dots, Q^k \in \mathcal{P}_{N'}$ be the paths whose existence is guaranteed by Proposition 4.6.12. We show that $Q^1, \dots, Q^k$ are the certificate of C at N', concluding the proof. Fix $j \in [k]$. The parity and level of Q^j is the same as that of P^j. Hence, the parity of Q^j is π_j, and its level ℓ_j. Recall that Q^j is an S'-path. By part (ii) of Proposition 4.6.12, Q^j starts (resp. ends) at the same node as P^j when this belongs to S', and at some node of X' otherwise. Using property 1 of Proposition 4.6.12, we deduce that Q^j is a S'-path that starts (resp. ends) at u_j (resp. v_j) when $u_j \neq \emptyset$ (resp. $v_j \neq \emptyset$), and at a node $x \in X'$ otherwise.

We are left to show that paths $\{Q^j\}_{j=1,\dots,k}$ are X'-disjoint. Suppose not, then there exists $v \in X'$, $j \neq j'$ such that $v \in Q^j \cap Q^{j'}$. Note that v cannot be an internal node of Q^j or $Q^{j'}$, since those paths are pairwise internally vertex-disjoint by construction. On the other hand, if v is an endpoint of both Q^j and $Q^{j'}$, then the statement either follows from the X-disjointness of Q^j and $Q^{j'}$, or by part (ii) of Proposition 4.6.12. □

Conclusion

In this dissertation, we focused on studying the computational complexity for three discrete optimization problems and to provide either tractable algorithms for the simpler instances or to provide approximation algorithms (or prophet inequalities) for the difficult instances. We provided NP-hardness proofs for many problems that have been open for many years.

In the first part of the thesis, we took significant steps towards a comprehensive understanding of the optimal ordering problem when the distributions involved have support on a constant number of points. We provided a strong hardness result that shows the problem is NP-hard even for a very special case of 3 point distributions. Subsequently, we closed the problem for 2-point distributions, as well as the said special case of 3-point distributions, by providing a polynomial time algorithm and an FPTAS respectively. We also provided insights on the impact of ordering by proving improved prophet inequalities.

There is much left to investigate. An open question is whether the FPTAS derived in Section 2.5 can be extended to k-point distributions for any constant k (we know from [44] and [52] that an EPTAS is possible). Our hardness result does not rule out the possibility of such an algorithm. We proved that for two-point distributions, the expected reward under optimal ordering is within a factor of 1.25 of the prophet's reward, thus improving the well-known prophet inequality for worst-case ordering (from factor 2 to 1.25). Can such a prophet inequality be proven for best ordering in general k-point distributions? Finally, an interesting direction is to conduct such an investigation into optimal ordering for other parametric forms of distributions.

In the second part of the thesis, we looked at the $k > 1$ case. We showed that finding the

optimal static ordering is NP-hard in the very simple case where all but one distribution has support on $\{0, 1\}$. Any simpler instances than this would be trivial (all $\{0, 1\}$ distributions, so any ordering is optimal). We complemented this with an FPTAS on two-point distributions in the $k = 2$ setting. We also provided some prophet inequalities for both static and dynamic orderings for two-point distributions.

There are many questions that are still left open. One question is whether there exists an FPTAS in the $k = 1$ case for general distributions. In addition, the best prophet inequality in the free order problem is a large open problem. For the $k > 1$ case, we don't know if finding the optimal dynamic policy for two-point distributions is tractable. Next, it would be good if we could extend our prophet inequalities from two-point distributions to the general distribution case.

In the last part of the thesis, we looked at the popular matching problem. We constructed a framework for mapping 3-SAT instances to bipartite graphs that helped us resolve many open complexity questions. Chief among them is the hardness of approximating a max weight popular matching to a factor better than $1/2$ and the hardness of finding a popular matching that is not stable nor dominant. We supplemented our hardness results with a tractable algorithm for popular matchings on bounded treewidth.

References

[1] Melika Abolhassani et al. “Beating 1-1/e for ordered prophets”. In: *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*. ACM. 2017, pp. 61–71.

[2] Saeed Alaei. “Bayesian combinatorial auctions: Expanding single buyer mechanisms to many buyers”. In: *SIAM Journal on Computing* 43.2 (2014), pp. 930–972.

[3] Ernst Althaus and Sarah Ziegler. “Optimal tree decompositions revisited: A simpler linear-time FPT algorithm”. In: *arXiv preprint arXiv:1912.09144* (2019).

[4] 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*. ACM. 2018, pp. 303–318.

[5] Hedyeh Beyhaghi et al. “Improved approximations for free-order prophets and second-price auctions”. In: *arXiv preprint arXiv:1807.03435* (2018).

[6] Péter Biró, Robert W Irving, and David F Manlove. “Popular matchings in the marriage and roommates problems”. In: *International Conference on Algorithms and Complexity*. Springer. 2010, pp. 97–108.

[7] Hans L Bodlaender. “A tourist guide through treewidth”. In: *Acta cybernetica* 11.1-2 (1994), p. 1.

[8] Hans Kleine Büning and Theodor Lettmann. *Propositional logic: deduction and algorithms*. Vol. 48. Cambridge University Press, 1999.

[9] Shuchi Chawla et al. “Multi-parameter mechanism design and sequential posted pricing”. In: *Proceedings of the forty-second ACM symposium on Theory of computing*. ACM. 2010, pp. 311–320.

[10] Jose Correa, Raimundo Saona, and Bruno Ziliotto. “Prophet secretary through blind strategies”. In: *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM. 2019, pp. 1946–1961.

[11] Jose Correa, Raimundo Saona, and Bruno Ziliotto. “Prophet secretary through blind strategies”. In: *Mathematical Programming* (2020), pp. 1–39.

[12] José Correa et al. “Posted price mechanisms for a random stream of customers”. In: *Proceedings of the 2017 ACM Conference on Economics and Computation*. ACM. 2017, pp. 169–186.

[13] Jose Correa et al. “Recent developments in prophet inequalities”. In: *ACM SIGecom Exchanges* 17.1 (2019), pp. 61–70.

[14] Bruno Courcelle. “The monadic second-order logic of graphs. I. Recognizable sets of finite graphs”. In: *Information and computation* 85.1 (1990), pp. 12–75.

[15] Ágnes Cseh. “Popular matchings”. In: *Trends in Computational Social Choice* 105.3 (2017).

[16] Ágnes Cseh and Telikepalli Kavitha. “Popular edges and dominant matchings”. In: *Mathematical Programming* 172.1-2 (2018), pp. 209–229.

[17] Agnes Cseh et al. “Understanding popular matchings via stable matchings”. In: *arXiv preprint arXiv:1811.06897* (2018).

[18] Reinhard Diestel. “Graph theory. 2005”. In: *Grad. Texts in Math* 101 (2005).

[19] Paul Dütting et al. “Prophet inequalities made easy: Stochastic optimization by pricing non-stochastic inputs”. In: *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE. 2017, pp. 540–551.

[20] Soheil Ehsani et al. “Prophet secretary for combinatorial auctions and matroids”. In: *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM. 2018, pp. 700–714.

[21] Hossein Esfandiari et al. “Prophet secretary”. In: *SIAM Journal on Discrete Mathematics* 31.3 (2017), pp. 1685–1701.

[22] Yuri Faenza, Vladlena Powers, and Xingyu Zhang. “Two-sided popular matchings in bipartite graphs with forbidden/forced elements and weights”. In: *arXiv preprint arXiv:1803.01478* (2018).

[23] Yuri Faenza et al. “Popular matchings and limits to tractability”. In: *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM. 2019, pp. 2790–2809.

[24] Moran Feldman, Ola Svensson, and Rico Zenklusen. “A simple O (log log (rank))-competitive algorithm for the matroid secretary problem”. In: *Proceedings of the twenty-sixth annual ACM-SIAM symposium on Discrete algorithms*. SIAM. 2014, pp. 1189–1201.

[25] Hao Fu, Jian Li, and Pan Xu. “A PTAS for a Class of Stochastic Dynamic Programs”. In: *arXiv preprint arXiv:1805.07742* (2018).

[26] David Gale and Lloyd S Shapley. “College admissions and the stability of marriage”. In: *The American Mathematical Monthly* 69.1 (1962), pp. 9–15.

[27] David Gilat. “On the best order of observation in optimal stopping problems”. In: *Journal of applied probability* 24.3 (1987), pp. 773–778.

[28] John Gittins, Kevin Glazebrook, and Richard Weber. *Multi-armed bandit allocation indices*. John Wiley & Sons, 2011.

[29] Sushmita Gupta et al. “Popular Matching in Roommates Setting is NP-hard”. In: *arXiv* (2018), arXiv–1803.

[30] Sushmita Gupta et al. “Popular matching in roommates setting is NP-hard”. In: *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM. 2019, pp. 2810–2822.

[31] Dan Gusfield and Robert W Irving. *The stable marriage problem: structure and algorithms*. MIT press, 1989.

[32] Mohammad Taghi Hajiaghayi, Robert Kleinberg, and Tuomas Sandholm. “Automated online mechanism design and prophet inequalities”. In: *AAAI*. Vol. 7. 2007, pp. 58–65.

[33] Theodore P Hill and Arie Hordijk. “Selection of order of observation in optimal stopping problems”. In: *Journal of applied probability* 22.1 (1985), pp. 177–184.

[34] Theodore P Hill and Robert P Kertz. “A survey of prophet inequalities in optimal stopping theory”. In: *Contemp. Math* 125 (1992), pp. 191–207.

[35] Theodore P Hill, Robert P Kertz, et al. “Comparisons of stop rule and supremum expectations of iid random variables”. In: *The Annals of Probability* 10.2 (1982), pp. 336–345.

[36] TP Hill. “Prophet inequalities and order selection in optimal stopping problems”. In: *Proceedings of the American Mathematical Society* 88.1 (1983), pp. 131–137.

[37] Mizuki Hirakawa et al. “On the structure of popular matchings in the stable marriage problem-who can join a popular matching”. In: *the 3rd International Workshop on Matching under Preferences (MATCH-UP)*. 2015.

[38] Chien-Chung Huang and Telikepalli Kavitha. “Popular matchings in the stable marriage problem”. In: *Information and Computation* 222 (2013), pp. 180–194.

[39] Telikepalli Kavitha. “A size-popularity tradeoff in the stable marriage problem”. In: *SIAM Journal on Computing* 43.1 (2014), pp. 52–71.

[40] Telikepalli Kavitha. “The popular roommates problem”. In: *arXiv preprint arXiv:1804.00141* (2018).

[41] Robert Kleinberg and Seth Matthew Weinberg. “Matroid prophet inequalities”. In: *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*. ACM. 2012, pp. 123–136.

[42] Ton Kloks. *Treewidth: computations and approximations*. Vol. 842. Springer Science & Business Media, 1994.

[43] Ulrich Krengel and Louis Sucheston. “Semiamarts and finite values”. In: *Bulletin of the American Mathematical Society* 83.4 (1977), pp. 745–747.

[44] Allen Liu et al. “Variable Decomposition for Prophet Inequalities and Optimal Ordering”. In: *arXiv preprint arXiv:2004.10163* (2020).

[45] Brendan Lucier. “An economic view of prophet inequalities”. In: *ACM SIGecom Exchanges* 16.1 (2017), pp. 24–47.

[46] Roger B Myerson. “Optimal auction design”. In: *Mathematics of operations research* 6.1 (1981), pp. 58–73.

[47] CT Ng et al. ““Product Partition” and related problems of scheduling and systems reliability: Computational complexity and approximation”. In: *European Journal of Operational Research* 207.2 (2010), pp. 601–604.

[48] Vladlena Powers. “Discrete Optimization Problems in Popular Matchings and Scheduling”. PhD thesis. Columbia University, 2020.

[49] 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*. ACM. 2016, pp. 324–332.

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

[51] Ester Samuel-Cahn et al. “Comparison of threshold stop rules and maximum for independent nonnegative random variables”. In: *the Annals of Probability* 12.4 (1984), pp. 1213–1216.

[52] 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*. 2021, pp. 793–794.

[53] Martin L Weitzman. “Optimal search for the best alternative”. In: *Econometrica: Journal of the Econometric Society* (1979), pp. 641–654.

[54] Qiqi Yan. “Mechanism design via correlation gap”. In: *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms*. Society for Industrial and Applied Mathematics. 2011, pp. 710–719.

Appendix A: Appendix to Chapter 2

A.1 Missing proofs from Section 2.3

Claim A.1.1. *Suppose $X_1, X_2, \ldots, X_n$ is an optimal ordering of the random variables. In an optimal stopping rule for this ordering, let S be the set of random variables that are accepted only when their realization is* 1*, and T be the random variables that are accepted whenever their realization is positive. Then, S precedes T. That is, $i < j$ whenever $X_i \in S$ and $X_j \in T$.*

Proof. Proof: Suppose there is an optimal ordering for which S does not precede T. In such a case there must be a pair of *adjacent* random variables X_i, X_j in the ordering such that $X_i \in S$, $X_j \in T$, and X_j appears before X_i. Let L be the sequence of random variables that precede X_j and R be the sequence of random variables that succeed X_i. We prove the claim by a standard interchange argument in which X_i and X_j are swapped: not surprisingly, the contributions from L and from R will be the same in both sequences, so their difference will assume a simple form.

Let σ_{ij} be the ordering where X_i precedes X_j and let σ_{ji} be the interchanged ordering where X_j precedes X_i. For their fixed thresholds, let $f(L)$ be the probability that none of the random variables in L is accepted and let $E(L)$ be the expected reward given that a random variable in L is accepted (similarly define for R). Define $V_{\sigma_{ij}}$ and $V_{\sigma_{ji}}$ to be the expected rewards under orderings σ_{ij} and σ_{ji} respectively, and under these fixed thresholds. Then, it is easy to verify that

$$V_{\sigma_{ij}} = E(L)(1-f(L)) + f(L)[q_i + (1-q_i)(p_j m_j + q_j) + (1-q_i)(1-q_j-p_j)E(R)(1-f(R))]$$

and

$$V_{\sigma_{ji}} = E(L)(1-f(L)) + f(L)[p_j m_j + q_j + (1-q_j-p_j)q_i + (1-q_j-p_j)(1-q_i)E(R)(1-f(R))].$$

Simplifying, we have:

$$V_{\sigma_{ij}} - V_{\sigma_{ji}} \;=\; f(L)p_j q_i(1-m_j) \;> 0.$$

Thus, swapping i and j while retaining their acceptance thresholds improves the original ordering, which, therefore, cannot be optimal. □

Claim 2.3.1. *Given random variables $X_1, \ldots, X_n$ that have three-point distributions with support $\{0, m_i, 1\}$,, and probabilities $\{1-p_i-q_i, p_i, q_i\}$ such that $m_i \in (0,1), p_i > 0, q_i > 0, p_i + q_i < 1$, for $i = 1, \ldots, n$. Then, an ordering σ is optimal only if (i) S^σ precedes T^σ; (ii) the random variables in S^σ are arranged arbitrarily; and (iii) the random variables in T^σ appear in a weakly descending order of E_i. In particular, if $E_1 = E_2 = \ldots = E_n$, then the random variables in T^{σ^*} can be arranged arbitrarily as well. Further, for any ordering σ satisfying (i)-(iii), if the unordered sets S^σ, T^σ are same as the unordered sets $S^{\sigma^*}, T^{\sigma^*}$ for some optimal ordering σ^*, then σ is also optimal.*

Proof. Proof: We already know that any ordering in which S does not precede T is sub-optimal, verifying (i). To see (ii), note that any random variable in S that is accepted results in a value of 1, and that the probability of accepting *some* random variable in S is $1 - \Pi_{i:X_i \in S}(1-q_i)$, regardless of how these random variables are ordered. We can verify (iii) using a simple interchange argument as well. Suppose $X_i, X_j \in T$. As before, we let σ_{ij} be an ordering in which X_i appears immediately before X_j, with L being the sequence of random variables that precede X_i and R being the sequence of random variables that succeed X_j. σ_{ji} is the ordering with X_i and X_j interchanged. Let $f(L)$ be the probability that none of the random variables in L is accepted and let $E(L)$ be the expected reward given that a random variable in L is accepted (similarly define for R). Define $V_{\sigma_{ij}}$ and $V_{\sigma_{ji}}$ to be the expected rewards under orderings σ_{ij} and σ_{ji} respectively, and under these fixed thresholds. Then, it is easy to verify that

$$V_{\sigma_{ij}} \;=\; E(L)(1-f(L)) + f(L)[p_i m_i + q_i + (1-q_i-p_i)(p_j m_j + q_j) + (1-q_i-p_i)(1-q_j-p_j)E(R)(1-f(R))]$$

and

$$V_{\sigma_{ji}} = E(L)(1-f(L))+f(L)[p_j m_j+q_j+(1-q_j-p_j)(p_i m_i+q_i)+(1-q_j-p_j)(1-q_i-p_i)E(R)(1-f(R))].$$

Simplifying, we have:

$$\begin{aligned} V_{\sigma_{ij}} - V_{\sigma_{ji}} &= f(L)[(p_j+q_j)(p_i m_i+q_i) - (p_i+q_i)(p_j m_j+q_j)] \\ &= \frac{f(L)}{(p_i+q_i)(p_j+q_j)}(E_i - E_j). \end{aligned}$$

Thus, it is optimal for X_i to appear before X_j in T if $E_i > E_j$.

For the second part, to see that any ordering satisfying properties $(i) - (iii)$ must be optimal note that the value of any ordering satisfying all of these properties is identical and must be optimal since the unordered sets S^{σ}, T^{σ} are the same as the unordered sets $S^{\sigma^*}, T^{\sigma^*}$. □

A.2 Missing proofs for section 2.4.2

We prove the following statement to complete the proof of Theorem 3 in Section 2.4.2.

Lemma A.2.1.

$$\max(T1, T2, T3) \geq 0.8MAX$$

where MAX is as defined in (2.10) *and* $T1, T2, T3$ *are as defined in* (2.11).

Using some algebraic manipulations, we can equivalently express MAX defined in (2.10) as:

$$\begin{aligned} MAX &:= p^* b^* + (1-p^*)p_w b_w + (1-p^*)(1-p_w)a^* \\ &= p^* b^* + p_w b_w - p^* p_w b_w + p^* p_w b^* - p^* p_w b^* + (1-p^*)(1-p_w)a^* \\ &= p^* p_w (b^* - b_w) + (1-p_w)p^* b^* + p_w b_w + (1-p^*)(1-p_w)a^* \\ &= f(p^*, p_w, b^*, b_w) + g(p^*, p_w, b^*, b_w) + h(p^*, p_w, a^*) \end{aligned}$$

where

$$f(p^*, p_w, b^*, b_w) := p^* p_w (b^* - b_w)$$

$$g(p^*, p_w, b^*, b_w) := (1 - p_w) p^* b^* + p_w b_w$$

$$h(p^*, p_w, a^*) := (1 - p^*)(1 - p_w) a^*$$

Now, notice that is $T1 = g + h$ and $T2 = f + g$. The idea of the proof is to bound the relative fraction of f or h to the offline expectation. We assume w.l.o.g. that $b_w = 1$. We can do this by multiplying every random variable by some appropriate constant α, such that $b_w = 1$. This scales the prophet's reward by α since $E[\max\{\alpha X, \alpha Y\}] = \alpha E[\max\{X, Y\}]$. Furthermore, $V(\alpha X, \alpha Y) = E[\max\{\alpha X, V(\alpha Y)\}] = \alpha E[\max\{X, V(Y)] = \alpha V(X, Y)$. Thus, the optimal reward also scales by α and so the competitive ratio remains the same.

The following three claims together prove the lemma statement.

Claim A.2.2. *If* $T1 \geq \max(T2, T3)$*, then* $T1 \geq 0.8MAX$

Proof. Proof of Claim A.2.2

We can assume that $\mu^* = p^* b^* + (1 - p^*) a^* \leq 1$ otherwise either $T2$ or $T3$ would be greater than T_1. Now since $T1 \geq T2$, this implies that $f \leq h$, and combined with the previous inequality, we get

$$\begin{aligned} p^* p_w (b^* - 1) &\leq (1 - p^*)(1 - p_w) a^* \\ &\leq (1 - p_w)(1 - p^* b^*) \end{aligned}$$

Rearranging terms, we get $p_w \leq \frac{1 - p^* b^*}{1 - p^*}$. Next, we will prove the following:

$$f(p^*, p_w, b^*) < \frac{1}{3} g(p^*, p_w, b^*)$$

The derivation is as follows:

$$\begin{aligned}
\frac{g(p^*, p_w, b^*)}{f(p^*, p_w, b^*)} &= \frac{(1-p_w)p^*b^* + p_w}{p^*p_w(b^*-1)} \\
&= \frac{(\frac{1}{p_w} - 1)p^*b^* + 1}{p^*(b^*-1)} \\
&\geq \frac{(\frac{p^*(b^*-1)}{1-p^*b^*})p^*b^* + 1}{p^*(b^*-1)} \\
&= \frac{(p^*)^2(b^*-1)b^* + 1 - p^*b^*}{p^*(b^*-1)(1-p^*b^*)} \\
&= \frac{p^*b^*}{1-p^*b^*} + \frac{1}{p^*(b^*-1)} \\
&= \frac{t}{1-t} + \frac{1}{t-p^*}
\end{aligned}$$

Where we let $t = p^*b^*$. For fixed p^*, this term is minimized when $t = \frac{1}{2}(1+p^*)$. Plugging this in, the minimum is $\frac{3+p^*}{1-p^*}$. This is minimized when $p^* \to 0$. Thus, $\frac{f(p^*,p_w,b^*)}{g(p^*,p_w,b^*)} < \frac{1}{3}$.

When $T1$ is maximum, it gives a competitive ratio of $\frac{g+h}{f+g+h} \geq \frac{g+f}{g+2f} \geq \frac{(4/3)g}{(5/3)g} = 4/5$. □

Claim A.2.3. *If* $T2 \geq \max(T1, T3)$*, then* $T2 \geq 0.8MAX$

Proof. Proof:

We will prove that

$$h(p^*, p_w, a^*) \leq \frac{1}{3}g(p^*, p_w, b^*)$$

Notice that since $T2 \geq T3$, we must have $a^* \leq p_w$ and $b^* \geq 1$. Furthermore, $h \leq f$ so combining these inequalities, we have the relation $(1-p^*)(1-p_w)a^* \leq (1-p^*)(1-p_w)p_w \leq p_w p^*(b^*-1) \Rightarrow b^* \geq \frac{(1-p^*)(1-p_w)}{p^*} + 1$. Using these two inequalities,

$$\begin{aligned}\frac{g(p^*, p_w, b^*)}{h(p^*, p_w, a^*)} &= \frac{(1-p_w)p^*b^* + p_w}{(1-p^*)(1-p_w)a^*} \\ &\geq \frac{(1-p_w)^2(1-p^*) + (1-p_w)p^* + p_w}{(1-p^*)(1-p_w)p_w} \\ &= \frac{1-(1-p^*)(1-p_w)p^*}{(1-p^*)(1-p_w)p_w} \\ &= \frac{1}{(1-p^*)(1-p_w)p_w} - \frac{p^*}{p_w}\end{aligned}$$

Differentiating this with respect to p^*, we see that the minimum is when $p^* = 1 - p_w$. Putting this back in, we have to minimize the term

$$\frac{1}{(1-p_w)^2 p_w} - \frac{1-p_w}{p_w}$$

The minimum is 3, attained at $p_w = 0$, proving that $h(p^*, p_w, a) \leq \frac{1}{3}g(p^*, p_w, b^*)$.

When $T2$ is maximum, it gives a competitive ratio of $\frac{f+g}{f+g+h} \geq \frac{g+h}{g+2h} \geq \frac{(4/3)g}{(5/3)g} = 4/5$. □

Claim A.2.4. *If* $T3 \geq \max(T1, T2)$*, then* $T3 \geq 0.8MAX$.

Proof. Proof:

$T3$ has an expected reward of $p^*b^* + (1-p^*)a^*$. Since $T3 \geq T2$, we have $a^* \geq p_w$, and since $T3 \geq T1$, we have $p^*b^* + (1-p^*)a^* \geq p_w + (1-p_w)p^*b^* + (1-p_w)(1-p^*)a^*$. Now suppose we swap the variables a^* and p_w in the expected reward and in the constraints. Then the "expected reward" is

$$p^*b^* + (1-p^*)p_w$$

and we have the constraints

$$p_w \leq a^*$$

$$p^*b^* + (1-p^*)p_w \geq a^* + (1-a^*)p^*b^* + (1-a^*)(1-p^*)p_w$$

The second constraint can be rearranged to show that $b^* \geq \frac{(1-p^*)(1-p_w)}{p^*} + 1$. Thus, this case can be reduced to the case of Claim A.2.3, so the rest of the proof is the same. □

A.3 Analysis of Algorithm 4: Proof of Theorem 5

We show that Algorithm 4 is an FPTAS for the problem in (2.12) of finding an optimal ordered partition. Algorithm 4 discretizes the interval $[0, 1]$ using a multiplicative grid with parameter $1 - \frac{\epsilon}{2n}$, so that it needs to search over only $poly(n, 1/\epsilon)$ partitions. Lemma A.3.1 allows us to restrict to these grid points. Then, in Lemma A.3.2 and Lemma A.3.3, respectively, we show that Algorithm 4 achieves the required approximation and run-time, to complete the proof of Theorem 5.

Lemma A.3.1. *Let $\mathcal{L}' \subseteq \mathcal{L}$ be the set of all ordered partitions $\{(S', T')\}$ in $\mathcal{L}$ with additional restriction on the last variable X in (S, T) that $V(X) = E[\max(X, 0)] \geq \frac{\epsilon OPT}{2n}$ for $0 \leq \epsilon \leq 1$. Let $OPT' = \max_{(S',T') \in \mathcal{L}'} V(S', T')$. Then,*

$$OPT' \geq \left(1 - \frac{\epsilon}{2}\right) OPT.$$

Proof. Proof: Let $\delta := \frac{\epsilon \text{OPT}}{2n}$. W.l.o.g., assume that an optimal ordered partition (S, T) orders the variables as $(X_1, \ldots, X_n)$, so that $\text{OPT} = V(X_1, \ldots, X_n)$. Suppose that there exist $1 \leq k \leq n$ such that $V(X_k) > \delta$. Then,

$$\begin{aligned} \text{OPT}' \geq V(X_1, \ldots, X_k) &\geq V(X_1, \ldots, X_n) - \sum_{i=k+1}^{n} V(X_i) \\ &\geq \text{OPT} - (n-1)\delta \\ &\geq (1 - \frac{\epsilon}{2})\text{OPT}, \end{aligned}$$

where the second inequality followed from repeatedly applying Lemma $A.4.2$. Now if such a k does not exist, then by Lemma A.4.2,

$$\text{OPT} = V(X_1, \ldots, X_n) \leq \sum_{i=1}^{n} V(X_i) \leq n\delta \leq \frac{\epsilon}{2}\text{OPT}$$

so that trivially, $(1 - \frac{\epsilon}{2})\text{OPT} \leq 0 \leq \text{OPT}'$.

□

Lemma A.3.2. *Let $\mathcal{L}^n$ be the set of ordered partitions returned by Algorithm 4 when run with parameters ϵ, MAX satisfying $\epsilon \in (0, 1)$ and $MAX \leq OPT$. And, let $ALG := \max_{(S,T)\in\mathcal{L}^n} V(S, T)$, Then,*

$$ALG \geq (1 - \frac{\epsilon}{2n})^n OPT',$$

where OPT' is as defined in Lemma A.3.1.

Proof. Proof: Let (S, T) be any ordered partition in $\mathcal{L}'$ where $\mathcal{L}' \subseteq \mathcal{L}$ is the restricted collection of ordered partitions defined in Lemma A.3.1 satisfying $V(X) = E[\max(X, 0)] \geq \frac{\epsilon\text{OPT}}{2n}$ for the last variable X in T (note that $T \neq \phi$ for all $(S, T) \in \mathcal{L}$). We show that there exist $(S', T') \in \mathcal{L}^n$ such that $V(S', T') \geq (1 - \frac{\epsilon}{2n})^n V(S, T)$.

We prove this by induction. Let (S_k, T_k) be an ordered partition obtained on restricting S, T to the first k variables $X_1, \ldots, X_k$ considered by the algorithm (here variables are ordered so that $E_1 \leq \cdots \leq E_k$). Let $1 \leq \bar{k} \leq n$ be such that $X_{\bar{k}}$ is the last variable in T and $V(X_{\bar{k}}) \geq \frac{\epsilon\text{OPT}}{2n}$; such a $\bar{k}$ must exist since $(S, T) \in \mathcal{L}'$ and $T \neq \phi$. We show that for all $k \geq \bar{k}$, at the end of the iteration k of the algorithm, there exists $(S'_k, T'_k) \in \mathcal{L}^k$ such that

$$\begin{aligned} V(S'_k) &\geq V(S_k), \\ V(T'_k) &\geq \rho^k V(T_k), \\ V(T'_k) &\geq \tfrac{\epsilon}{2n}\text{MAX}, \end{aligned} \tag{A.1}$$

where $\rho = (1 - \frac{\epsilon}{2n})$.

We prove the induction basis for $k = \bar{k}$. By definition of $\bar{k}$, $S_k = \{X_1, \ldots, X_{k-1}\}$ and $T_k = \{X_k\}$. Since $(\{X_1, \ldots, X_{k-1}\}, \phi) \in \mathcal{L}^{k-1}$, in the beginning of iteration k (before TRIM), the partition $(S''_k, T''_k) = (\{X_1, \ldots, X_{k-1}\}, \{X_k\})$ is added to $\mathcal{L}^k$. Since

$$V(T''_k) = V(X_k) \geq \tfrac{\epsilon}{2n}\text{OPT} \geq \tfrac{\epsilon}{2n}\text{MAX},$$

during TRIM this partition will fall in bucket $\mathcal{B}_j$ for some $j \geq 1$. By the trimming criteria, one partition (S'_k, T'_k) from bucket $\mathcal{B}_j$ will remain in $\mathcal{L}^k$ satisfying $V(T'_k) \geq \rho V(T''_k) = \rho V(T_k) \geq \rho^k V(T_k)$, $V(T'_k) \geq \rho^J max \geq \frac{\epsilon}{2n}\text{MAX}$, and $V(S'_k) \geq V(S''_k) \geq V(S_k)$. Therefore, S'_k, T'_k satisfies the conditions stated in (A.1) for $k = \bar{k}$.

Now, for the induction step, assume (A.1) holds for some $\bar{k} \leq k < n$. Then, in the beginning of iteration $k + 1$ (before TRIM is called), the algorithm will add two partitions $(\{X_{k+1}, S'_k\}, T'_k)$ and $(S'_k, \{X_{k+1}, T'_k\})$ to $\mathcal{L}^{k+1}$. We claim that one of these two partitions satisfies the required induction statement for $k + 1$, but with a better factor ρ^k instead of the required ρ^{k+1}. This can be observed as follows.

Depending on whether X_{k+1} appears in T_{k+1} or S_{k+1}, we can consider two cases: either $T_{k+1} = T_k$ or $S_{k+1} = S_k$. In the first case (when $T_{k+1} = T_k$), we set (S'_{k+1}, T'_{k+1}) as the first partition $(\{X_{k+1}, S'_k\}, T'_k)$. By induction hypothesis $V(T'_{k+1}) = V(T'_k) \geq \rho^k V(T_k) = \rho^k V(T_{k+1})$; also $V(T'_{k+1}) = V(T'_k) \geq \frac{\epsilon}{2n}\text{MAX}$; and

$$\begin{aligned} V(S'_{k+1}) = V(X_{k+1}, S'_k) &= V(X_{k+1}, V(S'_k)) \\ &\geq V(X_{k+1}, V(S_k)) \\ &= V(X_{k+1}, S_k) \\ &= V(S_{k+1}) \end{aligned} \tag{A.2}$$

where the second line follows from the induction hypothesiss.

For the second case (when $S_{k+1} = S_k$), we use the second partition, and set $(S'_{k+1}, T'_{k+1}) =$

$(S'_k, \{X_{k+1}, T'_k\})$, so that by induction hypothesis, $V(S'_{k+1}) = V(S'_k) \geq V(S_k) = V(S_{k+1})$, and

$$\begin{aligned} V(T'_{k+1}) = V(X_{k+1}, T'_k) &= V(X_{k+1}, V(T'_k)) \\ &\geq V(X_{k+1}, \rho^k V(T_k)) \\ &\geq \rho^k V(X_{k+1}, T_k) \\ &= \rho^k V(T_{k+1}) \end{aligned} \tag{A.3}$$

where the second line follows from the induction hypothesis and the third line follows from Lemma A.4.4. Also, $V(T'_{k+1}) = V(X_{k+1}, T'_k) \geq V(T'_k) \geq \frac{\epsilon}{2n}\text{MAX}$ by induction hypothesis.

However, one or both of these two partitions may be removed by the TRIM procedure. We claim that if any of the two partitions $(S'_{k+1}, T'_{k+1}) \in \{(\{X_{k+1}, S'_k\}, T'_k), (S'_k, \{X_{k+1}, T'_k\})\}$ is removed by the $TRIM$ procedure, then there will remain another partition (S''_{k+1}, T''_{k+1}) in $\mathcal{L}^{k+1}$ satisfying:

$$\begin{aligned} V(S''_{k+1}) &\geq V(S'_{k+1}), \\ V(T''_{k+1}) &\geq \rho V(T'_{k+1}), \\ V(T''_{k+1}) &\geq \tfrac{\epsilon}{2n}\text{MAX} \end{aligned} \tag{A.4}$$

To see (A.4), note that since $V(T'_{k+1}) \geq \frac{\epsilon}{2n}\text{MAX}$, (S'_{k+1}, T'_{k+1}) falls in a bucket $\mathcal{B}_j$, $j \neq 0$ during the TRIM procedure. Thus, the TRIM procedure will select one partition from this bucket, let it be (S''_{k+1}, T''_{k+1}). By definition of buckets, $V(T''_{k+1}) \geq \frac{\epsilon}{2n}\text{MAX}$. Also, by the criteria for selecting a partition from a bucket, we have $V(S''_{k+1}) \geq V(S'_{k+1})$, and by construction of buckets, if $j \neq 0$, $V(T''_{k+1}) \geq \rho V(T'_{k+1})$.

Together, (A.2), (A.3), (A.4) prove the induction statement in (A.1). Applying (A.1) for $k = n$, we get that there exists $(S'_n, T'_n) \in \mathcal{L}^n$ satisfying

$$
\begin{aligned}
V(S'_n, T'_n) &= V(S'_n, V(T'_n))) \\
&\geq V(S'_n, \rho^n V(T_n)) \\
&\geq V(S_n, \rho^n V(T_n)) \\
&\geq \rho^n V(S_n, V(T_n)) \\
&= \rho^n V(S, T)
\end{aligned}
$$

Here the first inequality followed from $V(T'_n) \geq \rho^n V(T_n)$. For the second inequality, note that a variable in S_n (and S'_n) is accepted if and only if it takes value 1. Therefore, S_n can be replaced by a $\{0, 1\}$ variable Y with probability $\prod_{i \in S_n} q_i$ to take value 1 (and similarly S'_n can be replaced by a $\{0, 1\}$ variable Y'). Then, since we have $E[Y'] = V(S'_n) \geq V(S_n) = E[Y]$, the second inequality follows from Lemma A.4.3. The third inequality follows from Lemma A.4.4.

This completes the proof of the lemma. □

Lemma A.3.3. *Algorithm 4 with parameters $\epsilon \in (0, 1)$ and MAX $\geq \frac{OPT}{2}$ runs in $O(\frac{n^4}{\epsilon^2})$ time.*

Proof. Proof: Given MAX $\geq \frac{\text{OPT}}{2}$, in the TRIM procedure (Algorithm 5), we always have $\frac{max}{\text{MAX}} \leq \frac{\text{OPT}}{\text{OPT}/2} \leq 2$. Therefore, the condition $\rho^J max \geq \frac{\epsilon}{2n}$MAX in the TRIM procedure ensures that the number of buckets

$$
J \leq \log_{1/\rho}(\frac{2n}{\epsilon}\frac{max}{\text{MAX}}) \leq \log_{1/\rho}(\frac{4n}{\epsilon}) = O(\frac{1}{(1-\rho)}\frac{n}{\epsilon}) = O(\frac{n^2}{\epsilon^2})
$$

Therefore, we maintain $O(\frac{n^2}{\epsilon^2})$ partitions in each iteration. Since for each partition, we need to calculate the expected reward, which is $O(n)$ time, and there are n iterations, we get the lemma statement. □

Now, we are ready to prove Theorem 5.

Proof. Proof of Theorem 5 Let $\mathcal{L}^n$ be the set of partitions returned by Algorithm 4 with parameters $\epsilon \in (0, 1)$, and

$$\text{MAX} := \tfrac{1}{2}E[\max(X_1, \ldots, X_n)].$$

Then, $\text{MAX} \geq \frac{1}{2}\text{OPT}$, so that by Lemma A.3.3, Algorithm 4 runs in time $O(\frac{n^4}{\epsilon^2})$ time. Also, using prophet inequality [51], $\text{MAX} \leq \text{OPT}$, so that by Lemma A.3.1 and Lemma A.3.2,

$$\text{ALG} \geq (1 - \epsilon/2)\text{OPT}' \geq (1 - \epsilon/2)^2\text{OPT} \geq (1 - \epsilon)\text{OPT}.$$

□

A.4 Other algebraic lemmas

We used following lemmas in the analysis.

Lemma A.4.1. ***Additive Scaling:*** *Given random variables $X_1, \ldots X_n$ and $c \in \mathbb{R}$ such that $Y_i := X_i + c$ is a non-negative random variable. Let σ be a permutation. Then $V(Y_{\sigma(1)}, \ldots, Y_{\sigma(n)}) = V(X_{\sigma(1)}, \ldots, X_{\sigma(n)}) + c$.*

Proof. Proof: We prove by induction. For one variable, $V(Y) = E[Y] = E[X+c] = E[\max\{X, 0\}]+ c = V(X) + c$. W.l.o.g, let $\sigma = (1, 2, \ldots, k + 1)$. For the inductive step:

$$\begin{aligned} V(Y_1, \ldots, Y_{k+1}) &= V(Y_1, V(Y_2 \ldots, Y_{k+1})) \\ &= V(X_1 + c, V(X_2 \ldots, X_{k+1}) + c) \\ &= E[\max(X_1, V(X_2, \ldots, X_{k+1})] + c \\ &= V(X_1, \ldots, X_{k+1}) + c \end{aligned}$$

where the second line follows from the induction hypothesis. □

Lemma A.4.2. *For any $v \geq 0$, $E[\max\{X, c + v\}] \leq E[\max\{X, c\}] + v$ and $E[\max\{X, c - v\}] \geq E[\max\{X, c\}] - v$*

Lemma A.4.3. *Let Y_1 and Y_2 be two $\{0,1\}$ random variables where $E[Y_1] \geq E[Y_2]$. Then $E[\max\{Y_1, c\}] \geq E[\max\{Y_2, c\}]$ for any constant $0 \leq c < 1$.*

Lemma A.4.4. *$E[\max\{X, \delta c\}]/E[\max\{X, c\}] \geq \delta$ for $0 \leq \delta \leq 1$*

Proof. Proof: For convenience we denote $V(X, c) := E[\max\{X, c\}]$

$$\begin{aligned}
\frac{V(X,\delta c)}{V(X,c)} &\geq \frac{V(X,c) - (c - \delta c)}{V(X,c)} \\
&= 1 - \frac{c(1-\delta)}{V(X,c)} \\
&\geq 1 - (1-\delta) \\
&= \delta
\end{aligned}$$

where in the first line, we used Lemma $A.4.2$ and in the third line, we used $V(X, c) \geq c$. □

Appendix B: Appendix to Chapter 4

B.1 Proof of Lemma 4.3.1

The fact that stable matchings are popular matchings of minimum size and that $V(S) \subseteq V(M)$ whenever S is stable and M is popular are known, see e.g. [39]. Hence, for any popular matching S' of minimum size, we have $V(S') \supseteq V(S)$ and $|S'| = |S|$, which implies $V(S') = V(S) \subseteq V(M)$ for any popular matching M, settling the first part. Now suppose that M again is popular, D is a dominant matching, and there exists $v \in V(M) \setminus V(D)$. Then, $G(M \triangle D)$ contains a path P starting at v with an edge of M, where we denote by $\triangle$ the symmetric difference operator. Hence, v is D-exposed. Since both matchings are popular, the number of nodes that prefer M to D in P is equal to the number of nodes that prefer D to M in P, else, either $M \triangle P$ would be more popular than M, or $D \triangle P$ would be more popular than D, a contradiction.

From this fact, we deduce the following. First, P has an even number of nodes. Hence, both the endpoints of P are D-exposed, and P is therefore D-augmenting in G. Second, if we label edges of $P \setminus D$ with "+" and "−" labels with respect to D as discussed at the beginning of Section 4.3, the total number of "+" and "−" labels on the edges of the path coincide. Hence, the number of $(+,+)$ edges and $(-,-)$ edges in P is the same. We consider two cases: if there is no $(+,+)$ edge, hence no $(-,-)$, the path is D-augmenting in G_D, contradicting the fact that D is dominant (see Theorem 14). Now suppose there is a $(+,+)$ edge e. Since the number of $(+,+)$ and $(-,-)$ edges in P coincide, there is a subpath P' of P satisfying one of the following: P' starts at a D-exposed node, traverses e, and does not contain a $(-,-)$ edge; or P' contains two $(+,+)$ edges and no $(-,-)$ edge. In both circumstances, we contradict Theorem 13: condition (ii) in the second case, and condition (iii) in the first.

B.2 Proof of claims from Section 4.4.5

Claim B.2.1. *If a positive clause contains a true literal ℓ, then $a(\ell), b(\ell) = (-,-)$. If a negative clause contains a true literal ℓ', then $e(\ell')f(\ell') = (-,-)$.*

Proof of claim. Consider $H(\ell)$ and $N(\ell')$. By hypothesis, $b(\ell)c(\ell), d(\ell)e(\ell) \in M$ and $c(\ell')e(\ell')$, $b(\ell')d(\ell') \in M$. By construction, $a(\ell), a(\ell')$ are matched to their favorite partners. This implies that $a(\ell)b(\ell) = (-,-)$ and $e(\ell')f(\ell') = (-,-)$. ◇

Claim B.2.2. *For any clause c, there is no malicious path starting at s that is contained in $\{s,t,u,v,w,x\} \cup \bigcup_{\ell \in c} V(\ell)$.*

Proof of claim. Since we have an assignment of the variables that satisfies ψ, then each clause contains at least one true literal. According to Claim B.2.1 $a(\ell)b(\ell) = (-,-)$ and $e(\ell')f(\ell') = (-,-)$ for some literals ℓ and ℓ' in positive and negative clauses correspondingly. The thesis follows since any malicious path that satisfies the hypothesis must pass through edges $a(\ell)b(\ell)$ for a positive clause c (or $e(\ell)f(\ell)$ for a negative clause) for all $\ell \in c$. ◇

Claim B.2.3. *A malicious path starting at $h(\ell)$ does not traverse $f(\ell)$. In particular, there is no malicious path from $h(\ell)$ to wx that is contained in $\{s,t,u,v,w,x\} \cup \bigcup_{\ell \in c} V(\ell)$ for any negative clause c.*

Proof of claim. $h(\ell)$ is unmatched and $e(\ell)f(\ell) \notin M$ regardless of the true or false assignment to literal ℓ. Hence P cannot take both $h(\ell)e(\ell)$ and $e(\ell)f(\ell)$, since none of them is in M. We conclude that there is no alternating path from $h(\ell)$ to wx contained only in c. ◇

Claim B.2.4. *Let ℓ be a positive literal, and let P be a malicious path. Then $c(\ell)c(\ell')$ is an edge of P for at most one negative literal ℓ' and if that happens, then $f(\ell) \notin P$.*

Proof of claim. The first statement follows from the definition of path. Now assume there is ℓ' such that $c(\ell)c(\ell') \in P$. We omit dependency on ℓ in nodes. By construction, $cc(\ell') \notin M$. Suppose by contradiction $f \in P$. Since f has degree 2, fd is an edge of P. We have two possibilities. First, if ℓ is true, then $bc, de \in M$. Then P or its inverse (i.e. the path obtained traversing nodes from P in opposite order) contains the subpath $f, d, e, c, c(\ell')$, a contradiction to the fact that P is M-alternating. Second, if ℓ is false, then $bd, ce \in M$. Then P or its inverse contains the subpath $f, d, b, c, c(\ell')$, again a contradiction to the fact that it is M-alternating. ◇

Claim B.2.5. *Let ℓ be a negative literal, and let P be a malicious path. Then $c(\ell)c(\ell')$ is an edge of P for at most one ℓ' and if that happens and $a(\ell) \neq u$, then $a(\ell) \notin P$.*

Proof of claim. Assume there is ℓ' such that $c(\ell)c(\ell') \in P$. Again, we omit explicit dependency on ℓ in nodes, and the first part of the statement is immediate. If $a \neq u$, then $\Gamma(a) = \{f(\ell''), b, c\}$ for the literal ℓ'' that precedes ℓ in its clause. We have two possibilities. First, if ℓ is true, then $ce, bd \in M$. This implies that P or its inverse contains the subpath $c(\ell'), c, e$. Hence if $a \in P$, then ac is not an edge of P, and consequently ab is an edge of P. Since $af(\ell'') \in M$ for some ℓ'' by construction and P is M-alternating, both ab and $bd \in M$ are edges of P. But then we must have that cd is an edge of P, a contradiction to P being a path. Second, if ℓ is false, then $cd, be \in M$. Hence, either P or its inverse contains the subpath $c(\ell')c, d, b, e$. Thus, ac, ab are not edges of P, which implies that $a \notin P$. ◇

B.3 Proofs from Section 4.6

B.3.1 Proof of Proposition 4.6.12

Proof. (iii). The proof of (i)-(ii) will only assume that M and N are (S, X)-locally popular matchings with the same (S, X)-tipping point. Recall that $N' = N \cup (M' \setminus M)$, $M' = M \cup (N' \setminus N)$ and that those unions are disjoint (by definition, Claim 4.6.7, and Claim 4.6.10). Therefore, the roles of M and N (hence those of M' and N') can be exchanged and the conclusions preserved. This

proves (iii) (assuming (i),(ii)).

(i)-(ii). Consider a sequence of paths $P_1, P_2, \ldots, P_q$ of G, $q \geq 2$, where, for $i \in [q-1]$, the last node of P_i coincide with the first node of P_{i+1}, and the paths are otherwise pairwise vertex-disjoint. The *concatenation* of $P_1, \ldots, P_q$ is the path obtained by listing the nodes of $P_1, P_2, \ldots, P_q$ in this order, and eliminating consecutive repeated nodes. If for $i \in [q-1]$, the last node of P_i coincide with the first node of P_{i+1}, and the paths are otherwise pairwise vertex-disjoint, with the exception of the last node of P_q coinciding with the first node of P_1, then the concatenation operation described above produces a cycle.

Let us now consider any path $P = (u_1, \ldots, u_v)$ from $P^1, \ldots, P^k$, where we omit the superscript for the sake of readability. We now write P as the concatenation of certain paths. We call this operation *path decomposition*. If $P \cap X = \emptyset$, we let $P_\infty := P$. Else, we write P as the concatenation of $P_0, P_1, P_2, \ldots, P_q, P_\infty$, iteratively constructed as follows:

- $P_0 = (u_1)$ if $u_1 \in X$; else, $P_0 = (u_1, \ldots, u_k)$ where k is the smallest index so that $u_{k+1} \in X$.
- For i odd, let u_j be the last node of P_{i-1}, and $t \geq j+1$ be the smallest index such that $u_t \in S$. Then $P_i = (u_j, \ldots, u_t)$.
- For i even, let u_j be the last node of P_{i-1}, and $t \geq j+1$ be the smallest index such that $u_{t+1} \in X$. Then $P_i = (u_j, \ldots, u_t)$.

Let q be the first index such that u_t as described above does not exists. If $u_{j+1} \in P$ (i.e., u_j is not the last node of P), set $P_\infty = (u_j, \ldots, u_v)$, else set $P_\infty = \emptyset$. See Figure B.1 for an example.

Note that some of the P_is (but not P_∞) could be trivial. Remove all non-trivial paths from the set $P_0, \ldots, P_q, P_\infty$ from the path decomposition, and note that this operation does not change the resulting P. Hence, in the following, we assume that all paths $P_0, P_1, P_2, \ldots, P_q, P_\infty$ (with possibly some subscripts being absent) from the path decomposition are non-trivial, and we let q be the subscript of the last non-trivial path (other than ∞).

The following claim collects some facts on $P_0, P_1, \ldots, P_q, P_\infty$ that easily follow by construction.

Figure B.1: On the left: a path P, starting from the top right (set X is in light gray, set S in dark gray). On the right: paths $P_0, P_1, P_2, P_3, P_\infty$ obtained by the path decomposition of P, starting from the top right, alternating between full and dashed lines.

Claim B.3.1. *Let $\{P_i\}_{i\in[q]\cup\{0,\infty\}}$ be the path decomposition of some path $P \in \{P^1, \ldots, P^k\}$ described above. Then:*

(i) *P is the path-concatenation of $P_0, P_1, \ldots, P_q, P_\infty$.*

(ii) *For all $i \in [q] \cup \{0, \infty\}$, $P_i \in \mathcal{P}_{M'}$.*

(iii) *For $i \in [q]$, $i \geq 2$, the first and last nodes of P_i belong to S.*

(iv) *For $i \in [q]$, i odd, P_i is an S-path and $P_i \subseteq X \cup S$.*

(v) *For $i \in [q]$, i even, $P_i \cap X = \emptyset$.*

(vi) *Assume $q \geq 1$. Then P_1 ends at a node of S, and starts either at a node of X, or at a node of S.*

(vii) *If $P_\infty \cap X \neq \emptyset$, then P_∞ is an S-path of $G_{M'}[X \cup S]$ ending at some node of X.*

Let us call a path P_i from a path decomposition *hidden* if (i is odd) or ($i = \infty$ and $P_\infty \cap X \neq \emptyset$ – i.e., condition (vii) in Claim B.3.1 is verified).

Claim B.3.2. *Consider the collection of paths $\{P_i^j\}_{j\in[k],i\in[q_j]\cup\{0,\infty\}}$, where for $j \in [k]$, $\{P_0^j, P_1^j, \ldots, P_{q_j}^j, P_\infty^j\}$ is the path decomposition of P^j. Then:*

(i) *$\{P_i^j\}_{j\in[k],i\in[q_j]\cup\{0,\infty\}}$ is a collection of pairwise internally vertex-disjoint paths and each node appears at most twice in the collection.*

(ii) The restriction of the collection $\{P_i^j\}_{j\in[k],i\in[q_j]\cup\{0,\infty\}}$ *to hidden paths is a family of pairwise* X*-disjoint* S*-paths from* $\mathcal{P}_M$*, and the level of each of those path is not* ∞*.*

(iii) Assume that $P_i^j \cap P_{i'}^{j'} \neq \emptyset$ *for some* $(i,j) \neq (i',j')$*, with* $i \leq i'$*. Then either (a)* $j = j'$*, the last node of* P_i^j *coincides with the first node of* $P_{i'}^j$ *and the two paths are otherwise disjoint, and (*$|i - i'| \leq 1$ *or* $i, i' \in \{q_j, \infty\}$*); or (b) the first node of* P_i^j *coincides with the last node of* $P_{i'}^j$*, and* i *(resp.* i'*) is the first (resp. last) path in the sequence of paths produced by the path decomposition operation on* P^j *(resp.* $P^{j'}$*).*

Proof of claim. (i). By Claim B.3.1, part (i), for $j \in [k]$, the concatenation of $P_0^j, P_1^j, \ldots, P_{q_j}^j, \ldots P_\infty^j$ gives P^j. By hypothesis, P^j, $j \in [k]$ form a collection of pairwise internally vertex-disjoint paths. The claim follows.

(ii). Restrict the collection $\{P_i^j\}$ to https://erc.europa.eu/document-libraryhidden paths. We already argued in Claim B.3.1, part (iv)-(vi) that those are S-paths. As they are subpaths of M'-alternating paths, they are also M'-alternating. Using again Claim B.3.1, part (iv)-(vi) they are contained in $X \cup S$. Using Claim 4.6.9, we deduce they are paths from $\mathcal{P}_M$. The fact that they are X-disjoint follows by hypothesis for $j \neq j'$, and by Claim B.3.1, part (i) for $j = j'$. The fact that M is (S', X')-locally popular implies that they cannot be of level ∞.

(iii). Let $P_i^j \cap P_{i'}^{j'} \neq \emptyset$, $(i,j) \neq (i',j')$. If $j = j'$, Claim B.3.1, part (i) implies that we are in case (a). Else, $j \neq j'$ and case (b) follows from the fact that P^j and $P^{j'}$ are internally vertex-disjoint. $\diamond$

Consider the ordered collection of paths

$$(P_i^j : j \in [k],\ i \in [q_j] :\ P_i^j \text{ is hidden})$$

and let u_i^j (resp. v_i^j) be the first (resp. last) node of each of those paths. Consider $C = (U, L, \Pi)$ defined as follows: $U = (U^1, \ldots, U^k)$, where U^j is the following ordered collection of pairs:

- (u_1^j, v_1^j) if $u_1^j \in S$, and $(\emptyset, v_1^j)$ otherwise.
- (u_i^j, v_i^j) for $3 \leq i \leq q_j$ odd and

- $(u^j_\infty, \emptyset)$ if P^j_∞ is hidden.

L is the ordered collection of levels ℓ^j_i of the hidden paths P^j_i , while Π is the ordered collection of their parities (both following the same order as the P^j_i). Note that $u^j_i \neq v^j_i$ because we restricted the path decomposition to non-trivial paths. Using Claim B.3.1, Claim B.3.2 and the hypothesis, one easily verifies the following.

Claim B.3.3. *C is an (S, X)-configuration. M is active at C, and the collection $\{P^j_i : j \in [k], i \in [q_j] : P^j_i$ is hidden$\}$ is a certificate of C at M.*

By hypothesis, N is also active at C. Hence, for all pairs (i, j) such that P^j_i is hidden, we can find pairwise X-disjoint S-paths $Q^j_i \in \mathcal{P}_N$, with each P^j_i starting (resp. ending) at node u^j_i (resp. v^j_i) if this belongs to S, and at a node of X otherwise, of the appropriate level and parity.

For $j \in [k]$, recall that P^j is the concatenation of $(P^j_0, P^j_1, \ldots, P^j_{q_j}, P^j_\infty)$, from which we removed the trivial paths. Consider the path Q^j obtained by replacing, in the concatenation above, each hidden path P^j_i with the corresponding Q^j_i. We conclude the proof of the proposition by showing that the collection of Q^j, $j \in [k]$, satisfies the claim.

Fix $j \in [k]$. Let us first argue that Q^j is a path in G that satisfies part (ii) from the statement of the proposition. Note that, by construction, for all hidden P^j_i, the first (resp. last) vertex of P^j_i coincides with the first (resp. last) vertex of Q^j_i, with possibly the exception of the first vertex of P^j_0 (resp. last vertex of P^j_∞) if it belongs to X. Moreover, since all Q^j_i are pairwise X-disjoint S-paths with no edge in $G \setminus X$, each Q^j is a path in G. The statement follows.

Let us now argue that Q^j is N'-alternating. First observe that $N'[(X' \cup S') \setminus X] = M'[(X' \cup S') \setminus X]$ follows by construction of N', Claim 4.6.8 and Claim 4.6.10. Moreover, all paths P^j_i that are not hidden are M'-alternating (by definition) and do not intersect X (by Claim B.3.1, part (v)). We conclude that all such P^j_i are also N'-alternating. For each hidden P^j_i, Q^j_i is an N-alternating path in $G_N[X \cup S]$ by construction. Since $G_N[X \cup S] = G_{N'}[X \cup S]$ by Claim 4.6.9, each Q^i_j is also N'-alternating. By construction, the parity of each hidden P^j_i coincides with that of the corresponding Q^j_i. We conclude that each Q^j is N'-alternating.

Let us now show that $Q^j \in \mathcal{P}_{N'}$, and it is of the same level and parity of P^j. Let P_i^j be non-hidden. By Claim B.3.1, part (iv), it is contained in $G[X' \cup S'] \setminus X$. By Claim 4.6.11, part (iii) and (iv), $P_i^j \cap N' = P_i^j \cap M'$, and each edge from $P_i^j \setminus N'$ has the same labels in $G_{M'}[X' \cup S']$ and $G_{N'}[X' \cup S']$. Now consider a hidden path P_i^j. By construction, Q_i^j is a path of $G_N[X \cup S] = G_{N'}[X \cup S]$ of the same parity and level of P_i^j. The statement follows.

We can now conclude that property (i) of the proposition holds, since:

$$\begin{aligned} Q^j \cap S' &= \cup_{i \in [q_j]:\, P_i^j \text{ non-hidden}} (P_i^j \cap S') \cup_{i \in [q_j]:P_i^j \text{ hidden}} (Q_i^j \cap S') \\ &= \cup_{i \in [q_j]:\, P_i^j \text{ non-hidden}} (P_i^j \cap S') \cup_{i \in [q_j]} (\{u_i^j, v_i^j\} \cap S') \\ &= P^j \cap S', \end{aligned}$$

where the first equality holds by construction; the second since $X \subseteq X'$ implies $S' \cap X = \emptyset$, and paths Q_i^j are by construction S-paths contained in $X \cup S$; the last, since by Claim B.3.1, part (iv) also hidden paths are S-paths contained in $X \cup S$.

Finally, let us observe that $Q^1, \ldots, Q^k$ are internally vertex-disjoint and X-disjoint. The X-disjointness follows from the fact that all Q_i^j corresponding to hidden P_i^j are X-disjoint, while other P_i^j do not intersect X by Claim B.3.1, part (v). The fact that $Q^1, \ldots, Q^k$ are pairwise internally vertex-disjoint follows from Claim B.3.2 and by definition of certificate. □

B.3.2 Proof of Claim 4.6.5

We start with an auxiliary claim.

Claim B.3.4. *For $i = 1, 2$, let $C_i = (U_i, L_i, \Pi_i)$ be a configuration at which M_i is active, and let $\mathcal{P}_i$ be a certificate of C_i at M_i. Then all paths from $\mathcal{P}_1, \mathcal{P}_2$ are pairwise internally vertex-disjoint.*

Proof of claim. By definition of certificate, paths from $\mathcal{P}_1$ are S_1-paths of $G[X_1 \cup S_1]$ that are X_1-disjoint. For $P \neq P' \in \mathcal{P}_1$, all internal nodes of P, P' belong to X_1 (by definition of S_1-paths). The definition of X_1-disjointness implies that they are internally vertex-disjoint. A similar argument

shows that any two paths from $\mathcal{S}_2$ are internally vertex-disjoint.

Now let $P_1 \in \mathcal{P}_1, P_2 \in \mathcal{P}_2$. By definition,

$$(X_1 \cup S_1) \cap (X_2 \cup S_2) = V(T_{B_1}) \cap V(T_{B_2}) = (V(T_{B_1}) \cap V(T_B)) \cap (V(T_{B_2}) \cap V(T_B)) = S_1 \cap S_2.$$

Hence, nodes of P_1 and P_2 can only intersect in $S_1 \cap S_2$. For $i = 1, 2$, by definition of S_i-path, no internal node of P_i belongs to $S_1 \cup S_2$, and the claim follows. $\diamond$

We now prove Claim 4.6.5. Assume first that there is a forbidden subgraph P (as in Theorem 13) in $G_{M^*}[X \cup S]$, whose existence certifies that M^* is not (S, X)-locally popular in G. Following the proof of Proposition 4.6.12, we can write P as the concatenation of $(P_1, P_2, \ldots, P_k)$ where each P_j is: either an S_i-path contained in $G[X_i \cup S_i]$ for $i \in \{1, 2\}$, or a path contained in $G[B]$. Construct $\mathcal{P}_1, \mathcal{P}_2$ as follows. Let $P' \in \{P_1, \ldots, P_k\}$. Then:

- if P' is an S_1-path contained in $G[X_1 \cup S_1]$, add it to $\mathcal{P}_1$;
- else if P' is an S_2-path contained in $G[X_2 \cup S_2]$, add it to $\mathcal{P}_2$.

Let $i \in \{1, 2\}$. By construction each $P' \in \mathcal{P}_i$ is a S_i-path contained in $G[X_1 \cup S_1] = G(V(T_{B_i}))$, and it is internally vertex-disjoint, hence X_i-disjoint, from all other paths from $\mathcal{P}_i$. Note that P' is also an M_1-alternating path of $G_{M_i}[X_i \cup S_i]$ (since $M^*_{X_i \cup S_i} = M_i$). Hence, since M_i is (S_i, X_i)-locally popular, we deduce that P' contains at most one $(+, +)$ edge. Hence, $\mathcal{P}_i$ forms the certificate of a certain (S_i, X_i)-configuration C_i at M_i.

Now consider the graph H and matching $M(H)$ obtained from configurations C_1 and C_2. Replace each path from $(P_1, \ldots, P_k)$ with the corresponding shortcut, and let Q be their concatenation. Then Q shows that $M(H)$ is not an $(S, V(H) \setminus S)$-locally popular matching in H, as required.

The opposite direction follows in a similar (inverse) fashion. Assume that, for $i = 1, 2$, there exists an (S_i, X_i)-configuration $C_i = (U_i, L_i, \Pi_i)$ at which M_i is active, such that, if we construct graph H and matching $M(H)$ as described, then $M(H)$ is not $(S, V(H) \setminus S)$-locally popular in H. Take such a pair of configurations that minimizes $|U_1| + |U_2|$. Let Q be the forbidden subgraph

(as in Theorem 13) in $H_{M(H)}$, whose existence certifies that $M(H)$ is not $(S, V(H) \setminus S)$-locally popular in G. For a pair $(u, v) \in U_1 \cup U_2$, denote by $H((u, v)))$ the subgraph of H induced by u, v and the $O(1)$ nodes added to H when pair (u, v) was considered in the construction of H. Write Q as the concatenation of paths $(Q_1, \ldots, Q_k)$, so that each Q_j is: either an $\{u, v\}$-path contained in $H((u, v))$ for some $(u, v) \in U_1 \cup U_2$; or a path contained in $H[B]$. By the choice of C_1, C_2, for each $(u, v) \in U_1 \cup U_2$, there exists exactly one Q_i that is contained in $H((u, v))$. Now replace each Q_j that is contained in some $H((u, v))$ for some $(u, v) \in U_i$ with the corresponding path in $G[X_i \cup S_i]$. By Claim B.3.4, the concatenation of this new sequence of path is a path, showing that M is not (S, X)-locally popular.