Stream sketches, sampling, and sabotage

by

Stephen R. Chestnut

A dissertation submitted to The Johns Hopkins University in conformity with the requirements for the degree of Doctor of Philosophy.

Baltimore, Maryland

February, 2015

Abstract

Exact solutions are unattainable for important problems. The calculations are limited by the memory of our computers and the length of time that we can wait for a solution. The field of approximation algorithms has grown to address this problem; it is practically important and theoretically fascinating. We address three questions along these lines. What are the limits of streaming computation? Can we efficiently compute the likelihood of a given network of relationships? How robust are the solutions to combinatorial optimization problems?

High speed network monitoring and rapid acquisition of scientific data require the development of space efficient algorithms. In these settings it is impractical or impossible to store all of the data, nonetheless the need for analyzing it persists. Typically, the goal is to compute some simple statistics on the input using sublinear, or even polylogarithmic, space. Our main contributions here are the complete classification of the space necessary for several types of statistics. Our sharpest results characterize the complexity in terms of the domain size *and* stream length. Furthermore, our algorithms are universal for their respective classes of statistics.

ABSTRACT

A network of relationships, for example friendships or species-habitat pairings, can often be represented as a binary contingency table, which is $\{0,1\}$-matrix with given row and column sums. A natural null model for hypothesis testing here is the uniform distribution on the set of binary contingency tables with the same line sums as the observation. However, exact calculation, asymptotic approximation, and even Monte-Carlo approximation of p-values are so-far practically unattainable for many interesting examples. This thesis presents two new algorithms for sampling contingency tables. One is a hybrid algorithm that combines elements of two previously known algorithms. It is intended to exploit certain properties of the margins that are observed in some data sets. Our other algorithm samples from a larger set of tables, but it has the advantage of being fast.

The robustness of a system can be assessed from optimal attack strategies. Interdiction problems ask about the worst-case impact of a limited change to an underlying optimization problem. Most interdiction problems are NP-hard, and furthermore, even designing efficient approximation algorithms that allow for estimating the order of magnitude of a worst-case impact has turned out to be very difficult. We suggest a general method to obtain pseudoapproximations for many interdiction problems.

Primary Reader: Rico Zenklusen

Secondary Reader: Vladimir Braverman

Acknowledgments

I am deeply indebted to my research advisers, Rico Zenklusen and Vladimir Braverman, for helping me get to this point. Rico's combinatorial optimization course captivated me, and towards the end of it we began working together on the sampling problem. The impact he has had on my mathematical confidence and ability since we met amazes me. I am still captivated. Rico, thank you for sharing your insights with me, I hope you will share more with me. I promise to do my best to keep up. I am grateful to Vova, for sharing so many problems with me, for his encouragement and honest criticisms, and for always pushing me to improve. I am still trying. I also appreciate his work building a community of computer science theorists at JHU; it has brightened my experience here to work and study with so many others.

I would like to thank Dr. James Fill for advising me for two years while I found my place at Hopkins and for teaching me precision in mathematics. I strive to hold myself to the same standards for clarity and precision in writing as he holds for himself. Donniell Fishkind has been a fantastic mentor to me; I can only hope to one day be such an engaging and capable teacher. Before I came to Hopkins, Manuel

Lladser was a great adviser to me and he put me on the track get here.

Quite a few people, besides those above, have contributed to the ideas in this dissertation. First, Carey Priebe suggested the contingency table sampling problem to me—I am thankful that to him for encouraging me, and I enjoyed my work on the problem, but I still don't have any samples for him. Youngser Park provided me with a collection of contingency tables derived from online forums. Daniel Sussman gave me another data set, useful discussions about the algorithm that became the Columns-In-Expectation sampler, and a lot of help with my Probability Theory homeworks.

Many other people at Hopkins have made the past four and a half years enjoyable and productive for me. In order of their appearance,...
Dan Naiman has been amazingly efficient at solving every Hopkins-problem I've had, even since the time I was only an applicant to the department. With John Wierman I've shared many helpful discussions on teaching and the academic's career. I will try to put them to good use. I thank Daniel Robinson and Amitabh Basu for teaching me optimization and for their friendly willingness to answer my questions. I am grateful to Tamás Budavári for supporting me in the last semester.

I am grateful Kristin Bechtel, Ann Gibbins, Heather Kelm, and Sandy Kirt for making my life so much easier with their friendly professionalism and attention to detail. The AMS Department is fortunate to be supported by such a capable staff.

My parents encouraged my curiosity and my interest in science and mathematics from a young age. Thank you for all of the science fairs and PTA meetings,

they worked.

Of course, I would be remiss not to thank my wonderful Juliana. You make my every day a dream. Thank you for sticking with me, I love you.

This work presented in this dissertation was partly supported by the U.S. Department of Education GAANN grant P200A090128, the National Science Foundation under Grant No. 1447639, a Campbell Fellowship, and a travel grant from the Acheson J. Duncan Fund for the Advancement of Research in Statistics.

Dedication

To Juliana and the rest of my family—Mom, Dad, Ted, Sara, Ellie, and Taylor.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Resources are limited but problems are not. The central challenge to theoretical computer scientists for 50 years has been efficient and accurate resource-limited computation.

Disregarding resources, it is often easy to describe an algorithm to solve a particular problem. For example, one can count the number of satisfying assignments to a Boolean formula with n variables simply by trying each of the 2^n possibilities. But for many problems, like SAT, it is not known whether there is any algorithm whose worst-case running time is not prohibitively large, or more precisely, bounded by a polynomial in the size of the input. What's worse is that it is generally believed that no such algorithm exists.

If we cannot solve a problem exactly with the time or storage that we have, what can we do? Often a "close" solution is good enough. Can we find one? This question is central to the study of approximation algorithms. Approximation is a formalization of heuristics. Our goal is to develop efficient algorithms for which we can *prove* that the approximate solutions they generate are close to the true solutions in the sense of some objective. We propose to address three problems with approximation algorithms.

First, large scale distributed computing and rapid data generation require us to re-evaluate

the definition of efficient computation. Data access times may be prohibitively large as data sets grow well beyond the memory capacity of any computer and even become geographically diverse. The streaming model of computation captures these challenges, and we study the approximatibility of so-called frequency functions in this model.

Second, the table of the Normal distribution in the back of a freshman statistics textbook does not cut it anymore with the advent of large-scale graphical data. New algorithmic and probabilistic tools are needed for statistical hypothesis testing on graphs and contingency tables, but in some respects they have been slow to develop. We propose a new algorithm for estimation in a commonly used null model: random bipartite graphs with a given degree sequence, or equivalently, random binary contingency tables. The model has been studied extensively, but current methods have either unacceptable running times or do not guarantee the accuracy of the results. Such a guarantee is crucial. Without one, who knows what null hypothesis is being tested!

Third, one way to understand the robustness of a system is to evaluate attack strategies. By studying interdiction, one might locate the most vulnerable edges in a transportation or communication network, find cost-effective strategies to prevent the spread of infection in a hospital [6], or determine how to inhibit the distribution of illegal drugs [96]. We address interdiction for a large class of structured combinatorial optimization problems. This class includes problems related to connectivity, matching, scheduling, network flows, and more.

1.1 Stream Sketches

Twitter has recorded 143,199 Tweets in one second [74]. On average last year, the world's largest public DNS was handling 1.5 million queries per second [50]. Sensors in the Large Hadron Collider record data 40 million times per second [27, p. 45]. The rate at which we generate new data is increasing so quickly that we cannot keep up if we want to store everything and process it exactly. Even problems that are traditionally considered easy by computer scientists, like finding the most

frequent item in a list, become essentially intractable for streaming data [3].

Distributed computing and rapid data acquisition drive the need for flexible statistics to stand in place of the data. They must require much less space to store and less time to query. The ultimate questions for us are: what statistics about a data stream can we compute if we are only capable of storing a tiny fraction of the stream? and what statistics can we not compute?

Theorists created the streaming model of computation to address these challenges. A stream S is a sequence of m integers $(s_1, s_2, \ldots, s_m)$ from the set $[n] = \{1, 2, \ldots, n\}$. Each stream has an associated frequency vector $f \in \mathbb{N}^n$ where the ith coordinate is the number of is that appear in the stream, formally $f_i = \#\{j \in [m] : s_j = i\}$. In the streaming model, the processor receives the elements of the stream once in an arbitrary order and the goal is to compute some function of the frequencies. The function depends on the application. For example, Lakhina et al. [76] propose using the entropy, $-\sum_i \frac{f_i}{m} \log(f_i/m)$, of an observed stream of IP packets as a statistic for anomaly detection in networks. Given a particular function g, we would like a randomized algorithm that, for any $\epsilon > 0$, can produce a $(1 \pm \epsilon)$-approximation to $\sum_i g(f_i)$, simply denoted $g(f)$, and that the space used by the algorithm is as small as possible. The problem is trivial with the memory to store f entirely. The question is: when can we get by with much less?

Space-optimal algorithms are known for approximating the frequency moments $F_k = \sum_{i=1}^{n} f_i^k$, for $k \geq 0$ (we use the convention that $0^0 = 0$) [23, 68, 69, 77]. A surprising result is that there is a polylogarithmic space $(1 \pm \epsilon)$-approximation algorithm for F_k when $k \leq 2$ [3, 56], but approximating F_k when $k > 2$ requires polynomial, more precisely $\Omega(n^{1-\frac{2}{k}})$, space [30]. Sublinear space approximation algorithms are also known for the entropy [28] of the stream $-\sum_i p_i \log p_i$, where $p_i = f_i/m$, for quantiles [38], and several functions.

The two main workhorses of these algorithms are *linear sketching* and *heavy elements algorithms*. To create a linear sketch, one selects a random $d \times n$ matrix A from a carefully chosen probability distribution and stores Af, often called the sketch. Since the transformation is linear, Af can be computed in one pass over the stream with $d \log m$ space. Finally, one uses a specialized

algorithm to extract an approximation to $g(f)$ from Af. It is an open problem to characterize the limits of linear sketching. Often, the specialized extraction algorithm takes from the sketch a list of heavy elements. A heavy element is an item $i \in [n]$ whose frequency f_i makes a large impact on $g(f)$, maybe it contributes 1% of the total value. In many scenarios, if one can find all of the heavy elements in a stream then one can approximate $g(f)$.

This thesis develops a new technique to derive nearly optimal universal sketching algorithms for decreasing g and two-pass heavy elements algorithms for increasing g. First, we take two known reductions used prove storage lower bounds and add a third. Next, we parameterize the reductions in terms of the frequency vector f. Consider the set of all streams with domain size at most n and length m, and let $\mathcal{F}$ be their frequency vectors. For each vector in $\mathcal{F}$, we have get a lower bound from the parameterized reduction. Obviously if $b(f)$ is the value of the lower bound derived from the frequency vector f, the best lower bound we can achieve with this reduction is $\max\{b(f) : f \in \mathcal{F}\}$. The result is that we get a lower bound for the streaming space complexity of approximating $g(f)$ that is expressed as the solution to a nonlinear optimization problem over the feasible set $\mathcal{F}$.

The optimality of the maximized lower bound plays a crucial role when we prove the correctness of our algorithms. What we use is the fact that which-ever is the stream that we receive as input we know that its frequency vector f has $b(f) \leq \max\{b(f') : f' \in \mathcal{F}\}$. Surprisingly, this information is all that we need in order to design a nearly space optimal sketching algorithm for decreasing functions and a heavy-elements algorithm for increasing functions. In fact an even stronger conclusion applies, the algorithms are *universal*. Indeed, suppose g_1 and g_2 are two functions with lower bound values $v_1 \leq v_2$, respectively. Then, as we will show, the sketch we use to approximate g_2 is also a correct sketching g_1. In fact, this implies even that the function we wish to approximate need not be chosen until after the sketch has been constructed, so long as we know the value of the optimal lower bound.

From a practical standpoint it may seem like our algorithms are limited by the tractability of the nonlinear optimization problem $\max\{b(f) : f \in \mathcal{F}\}$, but that is not the case. We show that

maximum can be approximated to within a small, constant factor using only polylogarithmically many evaluations of g, and the approximate value is enough for nearly space-optimal algorithms.

1.2 Sampling

Relationships are data. Users of a social networks are related by their interactions. Species are related to their habitats. Neurons are related to the muscles they control. Genetic diseases are related to the cellular processes they disrupt. Though different in composition, one statistical model is natural for all of these settings.

A network of relationships can often be represented as a binary contingency table, which is $\{0,1\}$-matrix with given row and column sums. Binary contingency tables play an important role in the statistical analysis of relationships. Ecologists, for example, represent species-habitat relations as binary contingency tables. When they observe variation among species the question is: Does the variation occur by chance or natural selection? A natural null model for hypothesis testing here is the uniform distribution on the set of binary contingency tables with the same row and column sums as the observation. Manly [78] cites the observed abundance of some species over others and the fact that some locations may be naturally better suited to a diverse ecosystem in support of this null model, as these factors do not reflect competitive pressure among the species. However, exact calculation and asymptotic approximation of p-values, even Monte-Carlo approximation, and even for very simple test statistics, are so-far practically unattainable for many interesting examples. One reason for this is that sampling contingency tables with given row and column sums has so far proved difficult. However, if one can sample from this distribution, then by averaging one can accurately estimate probabilities.

It turns out that the problems of sampling and of approximately counting are equivalent [62]. Formally, given two sequences $r = \{r_i\}_{i=1}^m$ and $c = \{c_i\}_{i=1}^n$, let $\Sigma_{r,c}$ denote the set of $m \times n$ binary matrices that have row sums r and column sums c. To answer the hypothesis testing

question it is sufficient to be able to answer: how large is $|\Sigma_{r,c}|$? or how can we sample a matrix at random from $\Sigma_{r,c}$?

The problem is not so easy, and two generalizations illustrate the point. If we fix the values of some table entries or allow entries to take any non-negative integer values, instead of just 0/1, then the new counting problem is among the hardest counting problems known; more precisely it is #P-complete[1] [94, 42]. With this perspective it is, perhaps, surprising that the complexity of computing $|\Sigma_{r,c}|$ is unknown.

Several approaches have been tried for sampling from $\Sigma_{r,c}$. The most successful algorithm from a theoretical point of view is due to Bezáková, Bhatnagar, and Vigoda [17]. It is a simulated annealing algorithm that can sample in polynomial time given any feasible set of margins. However, the running time bound for their sampling algorithm is $O((nm)^2 D^3 d_{\max} \log^4 nm)$, where $D = \sum_i r_i$ and $d_{\max} = \max\{\max_i r_i, \max_i c_i\}$. For dense matrices with $m = \theta(n)$, we can simplify the bound to $O(n^{11} \log^4 n)$. The algorithm is impractical even for small instances, and completely unusable for anything larger.

Other approaches that have been tried include a Markov chain [40, 70] (there has been limited success bounding the mixing time), dynamic programming [82] (the running time is typically exponential in the largest column margin), and Sequential Importance Sampling [33, 52] (which does not give an accuracy guarantee).

Our first new algorithm does not sample from $\Sigma_{r,c}$; rather, it samples from the larger set of binary tables with row margins r and any column margins. But, it has some desirable properties. First, the distribution of its samples is not the uniform distribution on this larger set. Rather, if C are the (random) margins of a table sampled with this algorithm then $EC = c$, the column margins are correct in expectation. Furthermore, conditionally given that $C = c'$, for any c', the sampling distribution is the uniform distribution on $\Sigma_{r,c'}$. Second, after an initialization step it produces

[1] This is the same complexity as counting the number of satisfying assignments for a given Boolean formula (i.e. SAT instance).

samples in $O(mn)$ time[2], which is clearly optimal. The initialization step requires nonlinear root-finding, but we prove that under a natural necessary condition there exists a unique root. We present an algorithm that approximates the root; in practice, one could use one of the many stable root-finding software packages that is freely available.

Our second algorithm is a hybrid algorithm that combines elements of two previously known samplers, Miller and Harrison's dynamic programming algorithm [82] and the Configuration Model rejection sampler [98]. The Configuration Model is a simple probability distribution on integer-valued contingency tables that is easy to sample. The samples always have the correct margins, but may not be binary. However, conditionally given that the sample is binary its distribution is the uniform distribution on $\Sigma_{r,c}$. To run the algorithm, begin with the $m \times n$ zero matrix and initialize each row and column by creating r_i tokens for row i and c_j tokens for column j. Next, repeatedly choose a row and column token at random and add a 1 to the corresponding matrix entry. If the matrix has only 0/1 entries when the tokens are exhausted, then it is a uniform sample from $\Sigma_{r,c}$. Otherwise, reject and start over. The rejection sampler does not work well when some of the margins are large because there is a very high probability of getting values greater than 1 in those rows and columns.

The strategy of our hybrid algorithm is to modify the Configuration Model so that rows with large margins are always assigned 0/1 elements. The idea is to use dynamic programming to sample rows with high margins and then switch to the Configuration Model to sample the rest of the rows, when it becomes advantageous. The switch must be done in a way that preserves uniformity and employs the strengths of each algorithm. We accomplish this by modifying Miller and Harrison's algorithm [82] once to account for the token process, and a second time to reduce the running time from exponential time to polynomial time at the expense of an extra rejection step.

It works as follows. A user chooses two parameters $p \leq m$ and $d \leq \max_j c_j$. A dynamic programming phase runs to sample the p rows with the largest margins according to the same

[2]This bound hides some assumptions about the complexity of arithmetic, without them the bound only increases to $O(mn^2)$.

distribution as the token process *conditioned* to have all 0/1 entries. Next the token process is used to sample the entries in the remaining $m - p$ rows. The modifications that we make limit the time complexity of the dynamic programming phase to about $O(pn^{d+2}m^2 + p^2n^{2d+4}\log^2 m)$ for initialization, which must be performed only once, and $O(mn + p^2n^2 lognlog^2 m)$ for each sample. The sampling distribution is exactly uniform, and we can choose d so that the algorithm runs within the time that we are willing to wait. The trade-off is that a smaller d makes for a higher rejection probability in the extra rejection step.

1.3 Sabotage

Imagine a business that owns many machines, perhaps of several different types, and leases them to customers. Each customer requires a particular number of machines and may have restrictions, for example on the number of machines of a particular type or on the city where the machines are located. The business owns extra machines in case of failures, but what is the worst case? More precisely, what is the least number of machines to fail that impacts the business's ability to meet its customers' demands? Alternatively, what is the worst impact that a particular number of failures could have on the business's ability to serve its customers?

Such a problem is a combinatorial interdiction problem. One purpose of interdiction is to evaluate the robustness of a system. Typically, one begins with an optimization problem over a discrete set of feasible solutions (e.g. maximizing the number of machines currently in use) and some rules for modifying the feasible set (e.g. machines may fail and be removed from the available pool). The goal is to have as great an impact on the objective function as possible, subject to a budget constraint (e.g. at most, B machines may fail). The problem may be hard even though the original optimization problem is not.

Interdiction versions of several specific combinatorial optimization problems have received attention [41, 88, 96, 99]. We propose an approximation algorithm for a large class of structured

interdiction problems. The class of problems that we consider includes interdicting optimization problems related to connectivity, matching, scheduling, and network flows.

We propose a polynomial-time algorithm that finds two interdiction sets, R_1 and R_2, such that R_1 is over-budget and R_2 is under-budget. Furthermore, either R_1 is better than the optimum and uses less than twice the budget or R_2 satisfies the budget and is at least half as good as the optimum. To find R_1 and R_2, we begin with the min-max integer program in (1.1). The r variables in a solution will be the incidence vector of the optimal set of elements to interdict, i.e. $R = \{m \in M : r_m = 1\}$ will be a worst case set of machine failures.

$$\begin{aligned} \min_{r} \quad \max_{x} \quad & \textstyle\sum_{m \in M}(1 - r_m)x_m, \\ Ax &\leq b, \\ c^T r &\leq B, \\ r, x &\in \{0,1\}^M. \end{aligned} \tag{1.1}$$

Here is a description of the problem formulation. The constraints $Ax \leq b$ restrict x to be the incidence vector of a feasible set of leased machines. The objective function for this problem is modified by r. It counts the number of machines that are leased and have not failed. The second constraint expresses a general interdiction budget, e.g. to consider the case where at most B machines fail we would set $c_m = 1$, for all $m \in M$.

Solving this integer program directly is NP-hard [64]. Instead, we pursue the approach taken by Burch et al. [26] for network flow interdiction. We dualize and relax the integrality constraint we arrive at $\max\{D(\lambda) : \lambda \geq 0\}$ where $D(\lambda)$ is defined as

$$\begin{aligned} D(\lambda) := \min_{y,r} \quad & b \cdot y - \lambda(B - c \cdot r) \\ A^T y + r &\geq 1, \\ y, r &\geq 0, \end{aligned} \tag{1.2}$$

The sets R_1 and R_2, described earlier, can be found by interpreting its optimal solution.

It turns out that this technique works for a large class of interdiction problems with $\{0,1\}$-valued objective functions. If we specialize a bit to interdicting maximum independent sets in matroids, then our results even apply to objective functions with $\mathbb{N}$-valued coefficients and submodular interdiction cost functions. Submodular costs express a "buy in bulk" discount, and arise naturally in interdiction settings as they reflect cascading failures and collateral damage.

There is more structure left in (1.2) for us to exploit. The two solutions R_1 and R_2 correspond to adjacent vertices on the polytope $\{y : A^T y + r \leq 1; y, r \geq 0\}$. If the polytope is well-structured then it can be exploited to improve the guarantee. With this in mind, we apply our earlier results to the problem of b-stable set interdiction, which is NP-hard. The LP corresponding to (1.2) in this setting is particularly nice, and we exploit its adjacency structure to develop a PTAS for b-stable set interdiction.

Chapter 2

Monotone streaming sums

A stream is the input for computational problem, presented sequentially and in an arbitrary order. Streams describe computing scenarios where incremental updates arrive at a server or processor and the complete data are too large for the processor to store. Each update is a small change to a high dimensional vector f known as the *frequency vector*. A processor reads the updates one-at-a-time, without control of their order, and is tasked to compute a function on the frequency vector. For example, the function might be a norm or other summary statistic. The difficulty comes because the processor has too little memory to store the entire vector f. Algorithms using sublinear space, or even logarithmic space, in the length of f are needed.

Interest in streaming has exploded over the past twenty years. Streaming algorithms found early applications in database monitoring and IP traffic analysis—backbone internet routers see very high-throughput but have limited memory [84]. Another driver is rapid acquisition of scientific data. For example, the ATLAS sensor in the Large Hadron Collider at CERN generates about 70 petabytes of collision information per second [7]. It is impossible to save all of that with current technology, so some processing must occur on-the-fly without the data ever being stored. Finally, almost all known streaming algorithms are based on additive data structures that make streaming algorithms

very easy to implement for distributed computation. That makes streaming algorithms appealing for processing distributed data sets, including in-situ processing for distributed sensor networks. See references [8] and [84] for details on these applications and others.

This chapter addresses techniques for developing lower bounds and universal algorithms for streaming computation that can be applied to a wide range of functions. Specifically we study functions of the form $g(f) := \sum_{i=1}^{n} g(|f_i|)$ where $g : \mathbb{N} \to \mathbb{R}$ is monotone.[1] Precise definitions of the classes of functions that we study are given in Section 2.1.

The class of functions that we consider encompasses nearly all functions that have been studied before, including the frequency moments $g(x) = x^k$ and the entropy norm $g(x) = x \log x$. Most previous work is limited to the analysis of a lower bound and algorithm for a specific function g, whereas these results apply to large classes of functions. We characterize the space necessary to produce a $(1 \pm \epsilon)$-approximation to $g(f)$ in terms of the accuracy ϵ, the dimension n, and $\|f\|_1$, which is also the stream length in the insertion only model. We are the first to consider the precise dependence of the space complexity of streaming computations on $\|f\|_1$. The attention is warranted, we will show that if g is nonnegative and decreasing then the space complexity for any function approximating g may depend delicately on the relationship between n and $\|f\|_1$. Section 2.2 presents more detail about past work and our results.

Here is a rough outline of the rest of this chapter. Sections 2.1 and 2.2 formally define the problem, explore previous results, and explains our contribution in more detail. Section 2.3 describes an "archetypal sketch" that transforms the streaming approximation problem into the problem of randomly sampling the frequency vector. Included is a detailed description of an algorithm that samples from the (unknown) support of the frequency vector with storage at most $O(\log n)$ times that needed just to store the sample itself.

Next comes additional background on the heavy elements problem in Section 2.4, a further transformation of the sampling problem to that of finding individual elements, and reductions from

[1] This overloaded notation with g will persist throughout the chapter.

communication complexity for streaming lower bounds in Section 2.5.

Our main contributions are presented in Sections 2.6, 2.7, and 2.8. For the case where g is increasing or decreasing, we parameterize the lower bound reductions in terms of the coordinates of the frequency vector. This has the effect of giving us a huge collection of lower bounds, one for each frequency vector. Next, we find the best lower bound by maximizing the bound over the set of frequency vectors. The result is a storage lower bound that is parameterized by the dimension of f and its L^1 length and the bound is expressed as the solution to a nonlinear optimization problem.

Usually, one would next derive an algorithm, prove an upper bound on the storage that it requires, and then examine the gap between this value and the lower bound. In our case, the upper and lower bounds have a closer connection. It turns out that the optimality of the lower bound, the one we maximized over the set of available frequency vectors, is crucially important to prove the correctness of our approximation and heavy elements algorithms.

In fact, the algorithm only depends on the value of the lower bound and not on any other specifics of the function g. The consequence is that the approximation algorithm or heavy elements algorithm for a function g is also correct for every function in the same class (decreasing functions or increasing functions) that has the same or smaller space complexity as g. This property is called *universality* of the algorithm. Describing the lower bound as the solution to an optimization problem is the key to precisely understanding the dependence on $\|f\|_1$ and proving universality of the algorithm.

2.1 Preliminaries and assumptions

A *stream* is a sequence $S = ((d_1, \delta_1), (d_2, \delta_2), \ldots, (d_M, \delta_M))$, where $d_i \in [n]$ are called the items or elements in the stream and $\delta_i = \pm 1$ is an insertion or deletion of i. The frequency of $d \in [n]$ after $k \leq M$ updates is

$$f_d^{(k)} = \sum \{\delta_j | j \leq k, d_j = d\},$$

and the vector $f = f(S) = f^{(M)}$ is commonly referred to as the *frequency vector* of the stream S. We use $\overline{f} \in \mathbb{N}^n$ to denote $|f|$, the absolute value of f, with the coordinates reordered in decreasing order. We call the set $\text{supp}(f) = \{d \in [n] : f_d \neq 0\}$ is the *support* of the stream.

This model is commonly known as the *turnstile* streaming model, as opposed to the *insertion-only* model which has $\delta_i = 1$, for all i. The *strict-turnstile* model allows deletions, i.e. $\delta_i = -1$, but comes with the promise that $f^{(k)} \geq 0$, for all k.

Let $\mathcal{F} = \{f \in \mathbb{N}^n : \sum f_d \leq m\}$ and let $\mathcal{S}$ denote the set of streams S with $|f(S)| \in \mathcal{F}$ and at most M updates. The set $\mathcal{F}$ is the set of all nonnegative frequency vectors with L^1 norm at most m. Clearly, $\mathcal{F}$ is the image under coordinate-wise absolute value of the set of all frequency vectors with L^1 norm at most m. The set $\mathcal{S}$ should be taken to consist of all the turnstile streams unless otherwise stated. We assume $n \leq m$.

This chapter addresses the question: Given a function $g : \mathbb{N} \to \mathbb{R}$, how much storage is necessary for a streaming algorithm that approximates

$$g(f) = \sum_{d=1}^{n} g(|f_d|)$$

for the frequency vector f of any stream $S \in \mathcal{S}$?

A randomized algorithm $\mathcal{A}$ is a *streaming g-sum $(1 \pm \epsilon)$-approximation algorithm for $\mathcal{S}$* if

$$P\left((1-\epsilon)g(f) \leq \mathcal{A}(S) \leq (1+\epsilon)g(f)\right) \geq \frac{2}{3}$$

holds for every stream $S \in \mathcal{S}$. For brevity, we just call such algorithms "approximation algorithms" when g, ϵ, and $\mathcal{S}$ are clear from the context. We consider the maximum number of bits of storage used by the algorithm $\mathcal{A}$ over streams in $\mathcal{S}$ with worst case randomness.

All functions will be assumed to be nonnegative, unless it is explicitly stated otherwise. As we show in Section 2.6.1, sublinear space streaming sum approximation algorithms necessitate nonnegativity or nonpositivity, and if g has zeros, then it must be periodic. We will also assume

that $g(0) = 0$. We can extend g to a symmetric function with domain $\mathbb{Z}$ by defining $g(-x) = g(x)$, for $x > 0$, thus $\sum g(f_i) = \sum g(|f_i|)$. We will use this convention to avoid carrying around the $|\cdot|$ notation. The two main classes of functions that we study are

- "decreasing" functions $\mathcal{D}$, by which we mean nonnegative functions that are 0 at the origin and nonincreasing on the interval $[1, \infty)$ and
- "increasing" functions $\mathcal{I}$, which are 0 at the origin and nondecreasing on the interval $[0, \infty)$.

The assumption that $g(0) = 0$ is a natural one (indeed it is satisfied by virtually all of the previous work) since otherwise $g(f)$ depends on the specification of the domain. That is, for a fixed stream S the value of $g(f(S))$ will differ depending on how we choose n, even as the stream itself remains unchanged. That may be the desired behavior, but we will not address it here other than to say that our results do not apply and allowing $g(0) \neq 0$ makes a big difference, in general. Although it is restrictive to assume that g is symmetric, it is present in previous work on streaming sums including the frequency moments, and previous work on entropies assumes the strict turnstile model [53] or insertion only model [29, 28], which are even more restrictive than our symmetry assumption. It would be interesting to generalize our results to remove this assumption.

Summary data structures often used in streaming algorithms are called *sketches*. A sketch is a compressed, and possibly randomized, representation of the stream. The compression is not lossless, and sketches are designed with a specific application in mind. All of our algorithms are derived from sketches of the frequency vector f. In fact, our algorithms are based on *linear sketches*, which means that they take the form Af for a (short and fat) matrix A. Two prominent examples of linear sketches are CountSketch [32], which we discuss later, and the Johnson-Lindenstrauss transform [63].

Our algorithms assume a priori knowledge of m and n, where $m \geq \|f\|_1$ and $n \geq |\operatorname{supp}(f)|$. We assume that our algorithm has access to an oracle that computes g on any valid input. In particular, the final step of our algorithms is to submit a list of inputs (a sketch) for g. We do not

count the storage required to evaluate g or to store its value. It will sometimes be expedient to assume $g(1) = 1$, which is not restrictive because all of our results apply to multiplicative approximations.

2.2 Background and our results

Much of the effort dedicated to understanding streaming computation, so far, has been directed at the frequency moments $F_p = \sum |f_i|^p$, for $0 < p < \infty$, as well as F_0 and F_∞, the number of distinct elements and the maximum frequency respectively. In the turnstile model, F_0 is distinguished from $L_0 = |\operatorname{supp}(f)|$, the number of elements with a nonzero frequency.

The interest in the frequency moments began with the seminal paper of Alon, Matias, and Szegedy [3]. They present the first storage lower bound for any algorithm that approximates the frequency moments and a sublinear-space algorithm for approximating any moment. Their lower bound is $\Omega(n^{1-5/p})$ and it is based on the multiparty disjointness problem, which we describe in Section 2.5.1. Their algorithm is simple to describe and it achieves $O(n^{1-1/p})$, bits for $p > 2$. Given an insertion only stream of length m, they first choose an update $i \in [m]$ uniformly at random at then count the number of times that item appears in the remainder of the stream. If R is the count then $X = m(R^p - (R-1)^p)$ is an unbiased estimator of F_p. The final estimate comes by repeating the previous procedure, averaging to reduce the variance, and then taking a median of the averages to provide the accuracy guarantee. Alon, Matias, and Szegedy go on to describe a second algorithm that approximates F_2 in $O(\epsilon^{-2} \log n)$ bits. Two papers from Kane, Nelson, and Woodruff exactly characterize the space necessary to approximate F_p for $0 < p \leq 2$ and $p = 0$ as $\theta(\epsilon^{-2} \log(M) + \log\log n)$ [68] and $\theta(\epsilon^{-2} + \log n)$ [69], respectively. It turns out that the F_2 algorithm of Alon, Matias, and Szegedy is optimal.

For the case $p > 2$, the first improvements over the AMS algorithm, due to Coppersmith and Kumar [37] and Ganguly [46], pushed the storage needed down to $\tilde{O}(n^{1-1/(p-1)})$. Both of those papers take a similar approach to AMS wherein they define an unbiased estimator for F_p and use a

median-of-averages to drop the variance and provide the approximation guarantee.

A major shift in the design of streaming algorithms began the following year with the algorithm of Indyk and Woodruff [59] that solves the frequency moments problem with $n^{1-2/p}(\frac{1}{\epsilon}\log n)^{O(1)}$ bits. Their paper introduced a now popular recursive subsampling technique. The subsampling is done recursively by randomly discarding elements of $[n]$ at each step. The algorithm finds any so-called heavy elements in each of the sampled streams and produces its estimate from these heavy elements. A heavy element is a $d \in [n]$ that makes a significant contribution to F_p. Suppose that the frequency of d contributes at least $0 < \alpha < 1$ fraction of the total, i.e. $|f_d|^p \geq \alpha \sum_i |f_i|^p$. Then Hölder's Inequality implies that

$$f_d^2 \geq \alpha^{p/2} \frac{1}{n^{1-2/p}} \sum_i f_i^2,$$

which means that d is also a heavy element for F_2, albeit with the smaller heaviness parameter $\alpha^{p/2}/n^{1-2/p}$. From here, the Indyk and Woodruff employ the CountSketch data structure of Charikar, Chen, and Farach-Colton [32] to identify the heavy elements in $\tilde{O}(\epsilon^{-2}n^{1-2/p})$ space (this data structure is discussed in more depth in Section 2.4.1). The overhead for the subsampling is a multiplicative factor of $(\frac{1}{\epsilon}\log n)^{O(1)}$ on top of the storage for the CountSketch, so the total algorithm runs in $n^{1-2/p}(\frac{1}{\epsilon}\log n)^{O(1)}$ bits of storage. The main drawback to the recursive subsampling approach is that analyzing the algorithms is typically difficult.

Most of the algorithms to come after Indyk and Woodruff's paper use some variant of the recursive subsampling technique. Bhuvanagiri, Ganguly, Kesh, and Saha [19] reduced storage to $O(\epsilon^{-2-4/p}n^{1-2/p}\log^3 m)$ bits and then Braverman and Ostrovsky [24] and Andoni, Krauthgamer, and Onak[4] further reduced the polylogarithmic factors and dependence on ϵ. The best algorithms currently known are those of Ganguly [47], at $O(\epsilon^{-2}n^{1-2/p}\log n)$ bits in the turnstile model, and Braverman, Katzman, Seidell, and Vorsanger [23], at $O(n^{1-2/p})$ bits in the insertion-only model with $\epsilon = \Omega(1)$.

Improvements to the AMS lower bound for algorithms approximating F_p, when $p > 2$, pri-

marily came by improving the communication lower bound on the multi-party disjointness function. Its complexity was settled for one-way protocols, which implies a bound only on one-pass streaming streaming algorithms, by Chakrabarti, Khot, and Sun [30] giving a $\Omega(n^{1-2/p})$ bits lower bound for estimating F_p in the insertion only model. Of course, the lower bound also applies in the turnstile model because every insertion-only stream is also a turnstile stream. The communication complexity of multi-party disjointness was later settled for unrestricted protocols by Gronemeier [49], which implies the same bound on multi-pass streaming algorithms. Improvements by Li and Woodruff [77], using the communication complexity of a different function, and Andoni et al. [5], for linear sketches, bumped the frequency moments lower bound up to $\Omega(\epsilon^{-2}n^{1-2/k}\log n)$ for turnstile model algorithms. Thus the best algorithms known, those of [47] for the turnstile model and Braverman et al. [23] for the insertion-only model, match the lower bounds up to a constant factor (for some choices of ϵ).

Indyk and Woodruff's algorithm is also the main inspiration for our own heavy elements algorithm when $|x|^p$ is replaced by an increasing function $g \in \mathcal{I}$. More precisely, given the function g we derive a lower bound, call its value b, using the multi-party disjointness problem on the storage necessary for any streaming algorithm that approximates $g(f)$. Next, we show that if $g(\overline{f}_1) \geq \epsilon \sum_i g(f_i)$ then

$$\overline{f}_1^2 \geq \frac{1}{\tilde{O}(b)} \sum_i f_i^2,$$

which is to say that the largest frequency in the stream is $\tilde{\Omega}(1/b)$-heavy for F_2 if it is heavy for g, of course Hölder's Inequality no longer serves to prove the implication as it did for F_p. Thus, we can identify the heavy element d in $\tilde{O}(b)$ bits of space using a CountSketch. The details are in Section 2.8.

For a general function g not much is known about the space-complexity. Most research has focused on specific functions. Chakrabarti, Do Ba, and Muthukrishnan [29] and Chakrabarti, Cormode, and Muthukrishnan [28] sketch the Shannon Entropy, $g(x) = x \log x$, when m is sufficiently large compared to n. Both papers use an estimator like the AMS estimator for F_p, $p > 2$, and

take a median-of-averages approach. Harvey, Nelson, and Onak [53] approximate three entropies in polylogarithmic space: the Renyi $\log(\|f\|_\alpha^\alpha)/(1-\alpha)$, Tsallis $(1-\|x\|_\alpha^\alpha)/(\alpha-1)$, and Shannon entropies. They use interpolation theory to derive their estimates from approximations to the pth frequency moment, $0 < p \leq 2$.

Subsequently, Braverman and Ostrovsky [25] characterized nondecreasing functions that have polylogarithmic-space approximation algorithms, assuming $m = \text{poly}(n)$. This is the largest class of functions that has been characterized to date, and it represents the most progress on the general streaming sum problem thus far. They define a set of nondecreasing functions $\mathcal{T}$ and they show that $g \notin \mathcal{T}$ requires super-polylogarithmic space. The proof is again by a reduction from the multiparty disjointness problem. For functions in $\mathcal{T}$ they demonstrate an approximation algorithm based on the Indyk-Woodruff heavy elements technique.

Given two streams with frequency vectors $e, f \in \mathbb{N}^n$ such that $\|e\|_1 = \|f\|_1$, Guha, Indyk, McGregor [51] study the problem of sketching common information divergences between the streams, i.e. statistical distances between the probability distributions with p.m.f.s $e/\|e\|_1$ and $f/\|f\|_1$. A simplified version of their main result applies to "decomposable" distances, which have the form $\sum_i \phi(e_i, f_i)$, where $\phi : \mathbb{N} \times \mathbb{N} \to \mathbb{R}_{\geq 0}$ has $\phi(x, x) = 0$, for all x. In particular, they show that if there exist $a, b, c \in \mathbb{N}$ such that

$$\frac{\phi(a, a+c)}{\phi(b, b+c)} > \frac{1}{(1-\epsilon)^2},$$

then any $(1 \pm \epsilon)$-approximation algorithm for $\sum \phi(e_i, f_i)$ requires $\Omega(n)$ space. In their words [51], this "suggests that unless $\phi(e_i, f_i)$ is some function of $e_i - f_i$ then the distance is not sketchable." Their proof relies on a reduction from the communication complexity of disjointness. When $\phi(e_i, f_i)$ is a decreasing function of $e_i - f_i$ we almost exactly characterize the space complexity and when it is an increasing function of $e_i - f_i$ we describe a nearly optimal heavy elements algorithm and corresponding lower bound.

Another line of research that is relevant to this chapter relates to randomly sampling from

streams. It turns out that one can approximate $g(f)$ by sampling elements $d \in [n]$ with probability roughly $p_d = g(f_d)/\epsilon g(f)$ and then averaging and scaling appropriately (see the next section for more details). Our algorithms for nonmonotonic functions in Section 2.6 and for decreasing functions in Section 2.7 use this scheme. The same sampling problem has been considered before. It was first formalized by Monemizadeh and Woodruff [83] who then go on to focus on L_p sampling, which is sampling d proportionally to $|f_d|^p$ for some $p \in [0, 2]$. In follow-up work, Jowhari, Săglam, and Tardos offer L_p sampling algorithms with better space complexity [65]. Monemizadeh and Woodruff's algorithm uses a recursive subsampling scheme, much like the F_p estimation algorithm of Indyk and Woodruff. They afford their algorithm small multiplicative and additive error to the sampling probabilities, and this allows it to operate in polylogarithmic space. For our applications it is sufficient to bound only one side of the error, in particular we only require that the sampling probability of d is no less than $g(f_d)/g(f)$.

Another development along these lines is the "Precision Sampling" algorithm of Andoni, Krauthgamer, and Onak [4]. This algorithm has the advantage of being much simpler than most other F_p estimation algorithms because it avoids the recursive subsampling while still using nearly optimal space. The random sampling is done by randomly weighting the items in the stream and then finding heavy elements with a single CountSketch data structure.

There are also lower bounds for sampling algorithms, see especially the Ph.D. thesis of Bar-Yossef [10], although these algorithms are of a different sort than the ones we have just mentioned, in that they are only allowed to sample an elements of domain and learn their frequencies—they cannot, for example, use a CountSketch. Nonetheless, Bar-Yossef [10, p. 108] proves that the number of samples needed to estimate the frequency moment F_p is $\theta(n^{1-1/p})$ (specifically that is the complexity to achieve constant relative error with constant probability).

We should point out that although there are many online algorithms that address sampling an element $d \in [n]$ proportionally to a given set of nonnegative weights $a_d \geq 0$, for $d \in [n]$, they may not carry over easily to the streaming model. The reason is that in order to know the weight of an

item, in our case $a_d = g(f_d)$, we need to know its frequency, which is distributed over the stream. Deletions from the stream pose a further problem.

The problems we address have been open for some time despite on-going interest. Our results give partial answers to questions posed by Alon, Matias, and Szegedy [3] – which functions can be sketched effectively?, and Nelson [57] – which families of functions admit universal sketches?

Our goal is optimal universal sketches. We develop lower bounds on approximation algorithms for decreasing and increasing functions, a nearly optimal universal sketch for decreasing functions, and a universal heavy elements algorithm for increasing functions. In doing so we greatly expand the class of functions for which nearly optimal approximation or heavy elements algorithms and lower bounds are known. We parameterize the space complexity by ϵ, n, and m. As far as we know, this is the first characterization of a streaming sum space complexity that explicitly includes dependence on m. Indeed, as we argue at the end of the present section, dependence on m must be considered for generic streaming sums. Moreover, the space complexity can depend delicately on m, as it does for some of the functions we consider. In order to develop the algorithms with nearly matching bounds, we develop a new technique that directly links the upper and lower bounds. In particular, our main lower bounds are expressed as solutions to nonlinear optimization problems where the feasible set is a subset of $\mathcal{F}$. The optimality of the solution plays crucially into the correctness proofs of our algorithms. The values of the lower bounds are needed in order to implement our algorithms, so we also present efficiently computable constant factor approximation algorithms to the optimal solutions of the nonlinear problems.

Our main contribution is the classification of the space necessary, up to small factors, for approximating any decreasing function and of the space necessary for finding heavy elements for any increasing function. It is the first improvement on the complexity of general streaming frequency sums since 2010 [25]. Furthermore, each sketch is universal for the functions in its respective class with the same space complexity, and the sketches are linear. One consequence is that we extend the work of Guha, Indyk, and McGregor [51] mentioned at the end of the previous section by

characterizing the space necessary for sketching decomposable distances when $\phi(e_i, f_i)$ is a decreasing function of $e_i - f_i$.

Our storage bounds are parameterized in terms of the dimension n, L^1 length m, and approximation error ϵ. It is somewhat uncommon within the streaming literature to include dependence on m, but it is important for generic streaming sums. It is obvious that the space complexity is entirely independent of the value of $g(x)$ for $x > m$. Of course, those values can affect the space complexity if the frequency vectors are allowed to have length $m' > m$. This effect turns out not to impact the complexity of the frequency moments, as can be shown by adapting the multiparty disjointness lower-bound—the algorithms are already independent of the stream length.

It may be possible to apply our methods in order to parameterize according to length other than L^1, for example L^∞, or in terms of more general constraints on the set of feasible streams. The L^1 length has a practical advantage that it can be computed exactly for insertion-only and strict-turnstile streams by a one pass algorithm with at most $O(\log M)$ bits of memory, and a $(1 \pm \epsilon)$-approximation requires only $O(\epsilon^{-2} \log M)$ bits in the turnstile model [68].

2.3 An archetypal sketch

A random sample of the frequencies immediately comes to mind as a plausible sketch for streaming frequency sums. One would then compute the mean value of g among the sampled frequencies and scale the result to get an unbiased estimate to $g(f)$. Lets test this sketch with a thought experiment.

Suppose that we put each element of $\operatorname{supp}(f)$ in our sample independently with the same probability p (Section 2.3.1 describes one way to accomplish this efficiently), so that our sketch is a list of about $p|\operatorname{supp}(f)|$ items and their frequencies. How large must p for $g(f)$ to be well approximated? Consider $g(x) = x^2$ and two streams S and S' with frequency vectors $f = (1, 1, \ldots, 1) \in \mathbb{R}^n$ and $f' = (\sqrt{n}, 1, 1, \ldots, 1) \in \mathbb{R}^n$, respectively. One can check with Chebyshev's Inequality that $p = O(1/\epsilon^2 n)$ is sufficient for a $(1 \pm \epsilon)$-approximation to $g(f)$, with constant probability. That

means a sample of size $O(\epsilon^{-2})$ thus a total of $O(\epsilon^{-2} \log M)$ bits to store the frequencies that we sample.

For S', however, this almost certainly will not work when ϵ is moderate (say $\epsilon = 0.01$) and n is large. The problem is that we really need to know about that $\sqrt{n}$. If we are to have a decent chance of sampling it then we must have $p = \Omega(1)$—and a quick variance calculation shows that it cannot be smaller—but that makes for a much larger sketch! If we could find that $\sqrt{n}$ by another method and sample the rest of the frequencies as before, then we would be in business.

Still, the random sample is not far off, and we will see later that it works well for some types of functions. Let us get started with an "Archetypal Sketch" for generic, nonnegative streaming sums. Given any nonnegative function g, Proposition 1 loosely describes a sampling model that yields a sample of size $O(\epsilon^{-2})$. The important change is to sample each frequency proportionally to its contribution to the sum. Accomplishing the sampling with a streaming algorithm is a subject for the rest of the chapter.

Proposition 1. *(Archetypal Sketch) Let $X_d \sim$ Bernoulli(p_d) be pairwise independent random variables with $p_d \geq \min\left\{1, \frac{8g(f_d)}{\epsilon^2 g(f)}\right\}$, for all $d \in [n]$. Let $\hat{G} = \sum_{d=1}^{n} p_d^{-1} X_d g(f_d)$, then*

$$P(|\hat{G} - g(f)| \leq \epsilon g(f)) \geq \frac{7}{8}.$$

Proof. We have $E\hat{G} = g(f)$ and, by pairwise independence,

$$Var(\hat{G}) \leq \sum_d p_d^{-2} g(f_d)^2 Var(X_d) \leq \sum_d p_d^{-1} g(f_d)^2 = \sum_d \frac{1}{8}\epsilon^2 g(f) g(f_d) = \frac{1}{8}(\epsilon g(f))^2.$$

The result follows by Chebyshev's inequality. □

The main point of Proposition 1 is to reduce the streaming approximation problem to a streaming sampling problem. As we mentioned in the previous section, Monemizadeh and Woodruff [83] have already posed the problem of sampling $p_d \approx g(f_d)/g(f)$. Although, for our applications it is

enough that $p_d \geq g(f_d)/g(f)$.

By the way, the success probability of Proposition 1 can improved to any $1 - \delta$ by taking independently repeating the computation $O(\log \delta^{-1})$ times and taking the median value.

What can we do with the simple random sample sketch discussed above? What follows is a brief preview of what is to come in Theorem 14 and Corollary 15. Consider a stream with frequency vector f. We can trivially bound $g(f) \geq |\operatorname{supp}(f)| \min_{x \in [m]} g(x)$. Thus, if we let p_d, for all $d \in [n]$, be the minimum of 1 and

$$p = \frac{\max_{x \in [m]} g(x)}{\epsilon^2 |\operatorname{supp}(f)| \min_{x \in [m]} g(x)},$$

then sampling each element with this probability gives a good estimate by Proposition 1. The next section will describe how to do this without knowing $|\operatorname{supp}(f)|$ ahead of time. Recognize that the sketch is universal for approximating a streaming sum on any nonnegative function g' with

$$\frac{\max_{x \in [m]} g'(x)}{\min_{x \in [m]} g'(x)} \leq \frac{\max_{x \in [m]} g(x)}{\min_{x \in [m]} g(x)}.$$

Generally, this sketch is far from optimal, but the point here is to get some algorithm that always works and is sublinear for some choices of g. It does work pretty well for bounded functions with bounded reciprocals, i.e. when there exists $c > 0$ such that $c^{-1} \leq g(x) \leq c$ for all $x \geq 1$. As we will see later, it also works when g is periodic with period $\min\{x > 0 : g(x) = 0\}$.

The next section details data structures that implement the simple random sampling in the streaming setting with small space.

2.3.1 Sampling with small space

Motivated by the Archetypal Sketch, suppose we want to sample each item's frequency with probability approximately $s/|\operatorname{supp}(f)|$. There are two problems to overcome. First, we do not know $|\operatorname{supp}(f)|$, and thus the sampling probability, ahead of time. Second, the support may be much larger at some intermediate stage than in the end, and we do not know which are the elements of

$\operatorname{supp}(f)$ ahead of time. How can we accomplish the sampling with only $\tilde{O}(s)$ space?

The answer comes in Algorithm 3, it is not a new technique. Let us begin assuming that we know $p = s/|\operatorname{supp}(f)|$ and tackle the second problem, the algorithm will guess p, approximately. Select pairwise independent Bernoulli(p) random variables X_d, for $d \in [n]$, and disregard all elements with $X_d = 0$. Because $\{X_d\}_{d\in[n]}$ are pairwise independent, we only need $O(\log n) = O(\log n)$ bits to store them.

Still, the number of elements with $X_d = 1$ may be too large to store a counter for each, but we know that at the end of the stream many of their frequencies will be zero. Though, that may not be true at an intermediate stage in the stream. However, there is a well established technique used in the CountSketch of Charikar, Chen, and Farach-Colton [32] and the CountMinSketch of Cormode and Muthukrishnan [38] that does the job. The main ideas are

1. if we add many items to a bin and all but one of their final frequencies are 0, the total frequency of the bin is the single nonzero frequency, and

2. the identity of a random sample can be stored in small space if the sampling is done with limited independence.

Algorithm 1 TurnstileSketch data structure for counters.

procedure TurnstileSketch($S \in \mathcal{S}$, $s \in \mathbb{N}$, $r \in \mathbb{N}$)
  Independently sample r pairwise independent collections random variables $X_{i,d} \in [s]$, for $d \in [n]$, and $i \in [r]$
  $c, c' \leftarrow 0 \in \mathbb{R}^{r\times s}$
  for $(d, \delta) \in S$ **do**
    $c_{i,X_{i,d}} \leftarrow c_{i,X_{i,d}} + \delta$
    $c'_{i,X_{i,d}} \leftarrow c'_{i,X_{i,d}} + d\delta$
  end for
  return $c, c', (X_{i,d})_{i\in[r],j\in[n]}$
end procedure

The sketch is described in Algorithm 1, we label it the TurnstileSketch just to emphasize that it uses a different choice of parameters than the earlier works. The associated GetCounts, Algorithm 2, describes extraction of the nonzero frequencies from the data structure. GetCounts

differs from the extraction method of CountSketch, which uses a priority queue to continuously maintain a list of the elements, that would work here as well.

Algorithm 2 Extracting the counts from a TurnstileSketch.

procedure GetCounts($S \in \mathcal{S}$, $s \in \mathbb{N}$, $\delta > 0$)
  $r \leftarrow \lceil 24 \log(n/\delta) \rceil$
  $c, c', (X_{i,d})_{i\in[r],d\in[n]} \leftarrow$ TurnstileSketch$(S, 4s, r)$
  $n_d \leftarrow \#\{i : i \in [r], c_{i,X_{i,d}} \neq 0\}$, for all $d \in [n]$
  $U \leftarrow \{c'_{i,j}/c_{i,j} : i \in [4s], j \in [r]\} \cap \mathbb{Z}$
  if $|U| > s$ **then**
    return $\emptyset$
  else
    $F_d \leftarrow \text{mode}(c_{i,X_{i,d}})_{i\in[r]}$, for $d \in U$
    return $\{(d, F_d) : d \in U\}$
  end if
end procedure

Lemma 2. *Let $S \in \mathcal{S}$ be a stream with M updates and $|\operatorname{supp}(S)| \leq s$. If $r \geq 24 \log(n/\delta)$ then* GetCounts(S, s, δ) *correctly returns* $\operatorname{supp}(S)$ *with probability at least $1 - \delta$. The algorithm uses $O(rs \log M)$ bits of space.*

Proof. We first show that for every $d \in [n]$ the value of $\text{mode}(c_{i,x_{i,d}})_{i=1}^{r}$ is the frequency of d with sufficiently large probability. For d with nonzero frequency, it happens for most cells counting d because it is the only nonzero frequency in the cell. Thus, correctness for the counts also implies correctness for the labels.

Let $f = f(S)$ denote the true frequency vector of S, let $W = \{d : f_d \neq 0\}$. By construction, $P(X_{i,d} = X_{i,w}) \leq 1/4s$ for every $d \in [n]$ and $w \in W$ with $d \neq w$. If $X_{i,d} \neq X_{i,w}$, for all $w \in W$ then $c_{i,X_{i,d}}$ correctly contains f_d. This happens with probability at least $3/4$, for each i, because $|\operatorname{supp}(S)| \leq s$. Finally, Chernoff's bound implies

$$P(f_d \neq f'_d) \leq P\left(|\{i : c_{i,X_{i,d}} = f_d\}| \leq (1 - 1/3)\, 3r/4\right) \leq exp\left\{\frac{-(1/3)^2(3r/4)}{2}\right\} \leq \frac{\delta}{n}.$$

Hence, with probability at least $(1 - \delta)$ the algorithm correctly determines the frequency of every element in $[n]$.

At most $O(r \log n) = O(r \log n)$ bits are need to store the random variables and $O(sr \log M)$ bits are required to maintain both arrays of counters. At most $O(s(\log M + \log n))$ bits are required for U and F, so the space complexity is $O(sr \log M)$. □

The update time for the TurnstileSketch data structured build by Algorithm 1 is $O(r) = O(\log(n/\delta))$, assuming random access to the array. Extracting the counts with Algorithm 2 requires $O(sr)$ operations, but is only performed once.

Algorithm 3 Identically distributed sampling from $\text{supp}(f)$.

procedure SimpleSketch(Stream S, $s > 0$)
  $\ell \leftarrow \lceil \lg(n/s) \rceil$
  for $0 \leq i \leq \ell$ **do**
    Sample pairwise independent r.v.s $X_{i,d} \sim \text{Bernoulli}(2^{-i})$, for $d \in [n]$
    Let $S^{(i)}$ be the substream of S with items $\{d : X_{i,d} = 1\}$
    $U^{(i)} \leftarrow \text{GetCounts}(S^{(i)}, 96s, 1/48)$
  end for
  $L \leftarrow \widehat{L_0}(S^{(i)}, 1/8, 1/12)$ (Using the L_0 estimator of [69])
  $i^* \leftarrow \max\left\{0, \left\lceil \lg \frac{L}{18s} \right\rceil\right\}$
  return $U^{(i^*)}$
end procedure

Returning to the problem of sampling from the support of the stream, now we can remove the assumption that $|\text{supp}(f)|$ is known ahead of time and present the full sampling algorithm. The technique we use, including the improvement for insertion only streams, has been used before, for example by [69] for the distinct elements and L_0 estimation problems. The main idea is to run GetCounts $O(\log n)$ times, once for each guess of $|\text{supp}(f)| = 2, 4, 8, \ldots, n$, and figure out from the results which was the correct guess. When the guess is too large the sampling probability is too small and nearly all of the counters will be 0. On the other hand, if the guess is too small then the sampling probability is to large and nearly none of the counters will be 0. In between the extremes is the Goldilocks point where the sampling probability is correct. We know when we have hit it because there are neither too many nor too few 0s among the counters.

Theorem 3. *With probability at least* $3/4$*, Algorithm 3 samples each item in* $\text{supp}(f)$ *with probability* $p \geq s/|\text{supp}(f)|$ *and the resulting sample of size* $O(s)$*. The algorithm can be implemented with*

$O(s \log(M) \log^2(n))$ *bits of space.*

Proof. Let

$$k = \left\lfloor \lg \frac{|\operatorname{supp}(S)|}{16s} \right\rfloor.$$

If $i^* \in \{k-1, k\}$, streams $S^{(k-1)}$ and $S^{(k)}$ both have small enough support, and the two outputs $U^{(k-1)}$ and $U^{(k)}$ of GetCounts are correct, then the output is correct. We show that the intersection of these events occurs with probability at least 3/4.

First, with probability at least 11/12 L is $(1 \pm 1/8)$-approximation to $|\operatorname{supp}(S)|$. A direct calculations then shows that $i^* \in \{k-1, k\}$.

The following two inequalities arise from the definition of k

$$\frac{64s}{|\operatorname{supp}(S)|} \geq 2^{-(k-1)} \geq 2^{-k} \geq \frac{16s}{|\operatorname{supp}(S)|}. \tag{2.1}$$

The first inequality implies that the expected support sizes of $S^{(k-1)}$ and $S^{(k)}$ and their variances are all at most $64s$. Chebyshev's inequality implies that each of these values exceeds $96s$ with probability no larger than $64/32^2 = 1/16$. So long as they don't, both streams are valid inputs to GetCounts. The last inequality of (2.1), with Lemma 2, implies that the sampling probability is correct.

Putting it together, the total probability of failure is no larger than

$$\frac{1}{12} + \frac{2}{16} + \frac{2}{48} \leq \frac{1}{4}, \tag{2.2}$$

where the terms come from the $|\operatorname{supp}(S)|$ estimation, the support sizes of substreams $k-1$ and k, and GetCounts.

The space bound for turnstile streams follows from Lemma 2 and $\ell = O(\log n) = O(\log n)$ by assumption. Approximating the support size of the stream with $\widehat{L_0}$ can accomplished with $O(\log n \log\log m)$ bits using the algorithm of Kane, Nelson, and Woodruff [69]. □

Because of deletions in the turnstile model, we need to wait until the end of the stream to rule out any of the guesses of $|\operatorname{supp}(f)|$. This is not the case in the insertion only model. As soon as the number of nonzero counters grows too large we can infer that the sampling probability is too large and discard the sample. It turns out that doing so is enough to cut a $\log n$ factor from the space complexity of the SimpleSketch. A further $\log n$ factor can be saved because TurnstileSketch is not needed in the insertion model.

Corollary 4. SimpleSketch *can be implemented with* $O(s \log M + \log^2 n)$ *bits of storage for insertion-only streams.*

Proof. Define ℓ independent collections of pairwise independent random variables $Y_{i,d} \sim \text{Bernoulli}(1/2)$, for $d \in [n]$, and choose the random variables in the algorithm to be

$$X_{i,d} = \prod_{j=1}^{i} Y_{i,d}.$$

One easily checks that each collection $\{X_{i,d}\}_{d\in[n]}$ is pairwise independent and that $P(X_{i,d} = 1) = 2^{-i}$, for all i and d. Storing the seeds for the collection $Y_{i,d}$ requires $O(\log^2 n)$ bits.

We can first save a $\log n$ factor by bypassing GetCounts and instead simply storing counters for each element that appears in each of the ℓ substreams. The counters should be stored in a hash table or other data structure with no space overhead and a small look-up time. Let us label the maximum number of counters to be stored for each substream as t. We choose $t = \max\{96s, \ell\}$. If the set of counters for each substream is discarded as soon as the number of nonzero counters exceeds the limit of $O(t)$, then the total storage cannot grow to large.

According to Lemma 5, the algorithm uses more than $12t$ counters with probability at most $1/6\ell$, at any given instant.

For each $0 \le i \le \ell$ let $T^{(i)}$ be the longest prefix of stream $S^{(i)}$ such that $|\operatorname{supp}(T^{(i)})| \le s$ and let $k^{(i)}$ denote the number of updates in $T^{(i)}$. Now, notice that the number of counters stored locally maximum at each $k^{(i)}$ and increasing for updates between $k^{(i)}$ and $k^{(i+1)}$. Thus, it is sufficient

to bound the storage used by the algorithm at these points.

By a union bound, the probability that the number of counters used by the algorithm at any point $k^{(1)}, k^{(2)}, \ldots, k^{(\ell)}$ is more than $12t$ is at most $\ell \cdot 1/6\ell = 1/6$. Finally, adapting the final union bound of (2.2) in the previous proof we have that the probability of error is at most $(1/12) + (1/6) = 1/4$. □

Lemma 5. *Let $v \in \{0,1\}^n$, define ℓ independent collections of pairwise independent random variables $Y_{i,d} \sim$ Bernoulli$(1/2)$, for $s \in [n]$ and $i \in [\ell]$, and set*

$$X_{i,d} = \prod_{j=1}^{i} Y_{i,d}.$$

For a given $s \in \mathbb{N}$, set $k = 0$ if $\sum_d v_d \leq s$ or $k = \max\{i : v^T X_i > s\}$ otherwise, where $X_i = (X_{i,1}, X_{i,2}, \ldots, X_{i,n}) \in \{0,1\}^n$. Then

$$P(\sum_{i=k+1}^{\ell} v^T X_i > 4s) \leq \frac{1}{2s}.$$

Proof. The sum is clearly monotonically increasing, so without loss of generality assume $\ell = \infty$. Notice that if $k > 0$, the sum is unchanged (i.e. it remains the same random variable) upon replacing v with the coordinate-wise product of v and X_k. Thus we may also assume that $k = 0$, i.e. $|\operatorname{supp}(v)| \leq s$.

For each $d \in \operatorname{supp}(v)$, let $Z_d = \sup\{i : X_{i,d} = 1\}$. Notice that $\{Z_d\}_{d \in \operatorname{supp}(v)}$ is a pairwise independent collection of Geometric$(1/2)$ random variables and let $Z = \sum_{d \in \operatorname{supp}(v)} Z_d$. We have that

$$Z = \sum_{i=0}^{\infty} v^T X_i,$$

because $X_{i,d} = 0$ implies $X_{j,d} = 0$ for all $j > i$.

Pairwise independence implies $EZ = Var(Z) = 2|\operatorname{supp}(v)| \leq 2s$, and by Chebyshev's

inequality

$$P(|Z - 2s| > 2s) \leq \frac{Var(Z)}{4s^2} \leq \frac{1}{2s}.$$

□

Lets take stock of what we covered in this section. We began with the Archetypal Sketch, a rough outline for an approximation algorithm that essentially turned the streaming problem into a sampling problem. The Archetypal Sketch is "ready made" for universal algorithms because the sketch itself is just a (random) collection of the frequencies in the stream. Specifically, we do not apply the function until after reading the entire stream. If we have sampling probabilities p_d, for $d \in [n]$ and a class of functions $\mathcal{G}$ that each satisfy the hypothesis for Proposition 1 then the sketch is universal for $\mathcal{G}$—it provides the approximation guarantee for each function in the class.

The simplest choice for sampling probabilities is uniform with pairwise independence. We have shown how to implement uniform sampling in a streaming algorithm with low overhead on top of the size of the stored sample. Uniform sampling is typically not optimal, but there are cases where it works well. Some of these are discussed in Sections 2.6 and 2.7.

The next two sections present more background material. Section 2.4 discusses heavy elements algorithms, which are the foundation of the recursive subsampling scheme introduced by Indyk and Woodruff [59]. The idea formalizes a problem we ran into during our though experiment at the beginning of Section 2.3, namely, find an element that makes up a significant portion of $g(f)$ (if such an element exists).

Section 2.5 briefly introduces the font of lower bounds known as Communication Complexity. The only new material in this section is the DISJ+IND function and a lower bound on its communication complexity.

These disparate seeming parts come together in Sections 2.6 through 2.8. The section begins with some streaming algorithm space lower bounds by reduction from communication complexity. The lower bounds describe some aberrant behaviors leading to linear lower bounds. That is followed

by a general application of SimpleSketch and a specific example: approximating the geometric mean.

2.4 Heavy elements

A heavy elements algorithm finds the set of items the stream that contribute the most to the value of $g(S)$. Given a function g and a frequency vector $f \in \mathbb{Z}^n$ we call $i \in [n]$ an *α-heavy element* for g if

$$g(f_i) \geq \alpha g(f),$$

the value f_i is a α-heavy frequency. The set $H_g(\alpha, f) \subseteq [n]$ is the set of all α-heavy elements for g in f, and we simply write $H(\alpha, f)$ when the function g is clear from the context.

Two highly influential algorithms, CountSketch by Charikar, Chen, and Farach-Colton [32] and CountMinSketch by Cormode and Muthukrishnan [38], popularized the heavy elements problem. They find heavy elements for $g(x) = x^2$ and $g(x) = x$, respectively. What is more relevant to our problem is that first optimal (to within polylogarithmic factors) streaming approximation algorithm for the frequency moments F_p, when $p \geq 3$, uses CountSketch to find heavy elements for F_p. That algorithm is due to Indyk and Woodruff [59]. Subsequently, a similar approach is used by Braverman and Ostrovsky [25] in order to approximate $g(f)$ for functions owning polylogarithmic-space approximation algorithms.

Definition 6. A *(α, ϵ)-heavy elements algorithm for g* is an algorithm $\mathcal{H}(\mathcal{S}, \alpha, \epsilon, \delta)$ that for any stream in $\mathcal{S}$ returns a set of pairs $H = \{(i_j, w_j)\}$. With probability at least $1 - \delta$ the set H satisfies

1. $(1 - \epsilon)g(f_{i_j}) \leq w_j \leq (1 + \epsilon)g(f_{i_j})$, for all j, and
2. if i is α-heavy then $i = i_j$, for some j.

The algorithmic approaches of [59] and [25] is along the lines of the Archetypal Sketch. Let $g(x) = x^k$ and recall that the Archetypal Sketch calls for sampling each item d with probability

$p_d = \min\{1, \frac{8g(f_d)}{\epsilon^2 g(f)}\}$. To implement the sketch, one must find $\frac{8}{\epsilon^2}$-heavy elements with probability 1. Now, if $g(f_d) = 2^{-i}\frac{\epsilon^2}{8}g(f)$, for $i > 0$, then the Proposition 1 calls for sampling d with probability 2^{-i}. It turns out that this can be accomplished by randomly sampling a substream where each item appears with probability roughly 2^{-i} and then finding heavy elements in the substream. Indeed, this approach is used by Braverman and Ostrovsky [24] for the frequency moments. Their algorithm finds α-heavy elements, for $\alpha = \epsilon^2/\log^3 n$, in recursively subsampled streams. The strategy is to find all α-heavy elements, then randomly discard half of the elements in the stream and recurse[2].

The first application of this method, by Indyk and Woodruff [59], uses recursive subsampling and heavy elements in a slightly different manner. Their algorithm estimates

$$\#\{d \in [n] : \gamma^k \leq g(f_d) \leq \gamma^{k-1}\},$$

for $\gamma = 1 + \theta(\epsilon)$ and $k = 0, 1, 2, \ldots, O(\log n)$. The algorithm combines the estimates to approximate $g(f) = \sum_d |f_d|^k$.

The next section describes how CountSketch can be applied to find heavy elements for F_2. When we get to Section 2.8 we will reduce the problem of finding heavy elements for g, when g is increasing, to that of finding heavy elements for F_2 and then use CountSketch as a subroutine, mirroring the approach of Indyk and Woodruff.

2.4.1 CountSketch for F_2 heavy elements

The workhorse for previous streaming algorithms based on heavy elements is CountSketch for finding heavy elements. We will use it as well, so we quickly describe how to choose the parameters to get a (α, ϵ)-heavy elements algorithm for F_2.

CountSketch was introduced by Charikar, Chen, and Farach-Colton [32] to solve the problem $FindApproxTop(\ell, \epsilon)$. Recall that $\overline{f}_\ell$ is the magnitude of the ℓth largest frequency in f. The algorithm returns a set $\hat{H}$ of ℓ elements that approximates the set of items with the ℓ largest

[2]Of course, this simplified description omits many important details and technical legwork!

frequencies in the following way: with probability at least 2/3

- $\{s \in [n] : |f_s| \geq (1+\epsilon)\overline{f}_\ell\} \subseteq \hat{H}$ and
- $\{s \in [n] : |f_s| < (1-\epsilon)\overline{f}_\ell\} \cap \hat{H} = \emptyset$.

One can also extract from the sketch an additive $\pm\epsilon\overline{f}_\ell$ approximation to any coordinate of f, and thus one also gets a $(1 \pm \epsilon)$-approximation to the frequency of s when $|f_s| \geq \overline{f}_\ell$.

The sketch is a $r \times b$ array of counters with $r = O(\log n)$ and

$$b = O\left(\max\{\ell, (\epsilon\overline{f}_\ell)^{-2}\sum_{j>\ell}\overline{f}_j^2\}\right).$$

The absolute value of each counter is bounded by m, so the total space required by the algorithm is $O(br\log m)$.

Let us now show that the sketch can also be used as a (α, ϵ)-heavy elements algorithm requiring $O(\alpha^{-1}\epsilon^{-2}\log n \log m)$ bits of space. A priori, we do not know how many α-heavy elements a stream contains, but we can still determine the appropriate value of b. Suppose ℓ is the number of α-heavy elements in the stream. Then $\overline{f}_\ell^2 \geq \alpha F_2$, hence

$$\frac{1}{\overline{f}_\ell^2}\sum_{j>\ell}\overline{f}_j^2 \leq \frac{1}{\alpha F_2}\sum_{j>\ell}\overline{f}_j^2 \leq \frac{1}{\alpha}. \tag{2.3}$$

It also happens that $\ell \leq 1/\alpha$, so comparing with the definition of b we get $b = O(\epsilon^{-2}\alpha^{-1})$. It's easy to see that (2.3) can be sharp, so bound for b cannot be improved upon (using the analysis of [32]).

Replacing α with $(1-\epsilon)\alpha$ shifts the CountSketch guarantee so that

- $H_{\geq\alpha} \subseteq \hat{H}$ and
- $H_{\leq(1-\epsilon)^2\alpha} \cap \hat{H} = \emptyset$.

The sketch also yields a

$$\pm\sqrt{\frac{1}{b}\sum_{d=\ell+1}^{n}\overline{f}_d^2}$$

additive approximation to each frequency.

2.5 Communication Complexity

Reductions from communication problems are a central method for establishing lower bounds on the space complexity of streaming algorithms. The first such reductions were used by Alon, Matias, and Szegedy in their seminal paper addressing the frequency moments [3]. Since then, researchers have discovered optimal and nearly optimal lower bounds on the space complexity of many streaming problems with reductions from communication problems. The sequence of bounds [11, 30, 49] for the multi-party disjointness problem is perhaps the most relevant to this chapter, but there are many others. This section introduces some basic definitions that we will need in order to prove space complexity lower bounds on streaming algorithms. See the book by Kushilevitz and Nisan [75] for more background on communication complexity.

A communication problem is formulated as follows. There are $t \geq 2$ players that each have unlimited computational power. The players share a blackboard to communicate. Every player can read all of the bits written to the blackboard. There is a Boolean function $c : \mathcal{X} \to \{0, 1\}$, where $\mathcal{X} = \mathcal{X}_1 \times \mathcal{X}_2 \times \cdots \times \mathcal{X}_t$ for some sets $\mathcal{X}_1, \ldots, \mathcal{X}_t$, and player i is given an element $x_i \in \mathcal{X}_i$ but has no information about the other players' elements. The players can communicate with each other by writing bits to a shared blackboard and their goal is to compute $c(x_1, x_2, \ldots, x_t)$. The method of their communication is called a *protocol*. Formally, a protocol is a rooted binary tree where each leaf is labelled 0 or 1 and each non-leaf vertex v has two edges to its children labelled 0 and 1 and itself is labelled with a pair (i_v, b_v), where $i_v \in [t]$ is the name of the next player to write to the blackboard and $b_v : \mathcal{X}_{i_v} \to \{0, 1\}$ tells the bit that she will write. Each instance $x = (x_1, x_2, \ldots, x_t) \in \mathcal{X}$ identifies a path from the root to a leaf of the protocol by beginning with the root and recursively traversing from a vertex with label (i_v, b_v) to its $b_v(x_{i_v})$ child. We use $\mathcal{P}(x)$ to denote the label on the leaf identified by x. We say that $\mathcal{P}$ is a (correct) protocol for c if $\mathcal{P}(x) = c(x)$, for all $x \in \mathcal{X}$.

Given such a path $v_1, v_2, \ldots, v_\ell$ starting at the root v_1 and ending at a leaf v_ℓ, the sequence of bits $b_{v_1}(x_{i_{v_1}}), b_{v_2}(x_{i_{v_2}}), \ldots, b_{v_\ell}(x_{i_{v_\ell}})$ is called the *transcript* of the protocol $\mathcal{P}$ on the input x. We denote it $T_{\mathcal{P}}(x)$.

The *cost* of the protocol $\mathcal{P}$ is the height of the tree

$$|\mathcal{P}| := \max_{x \in \mathcal{X}} |T_{\mathcal{P}}(x)|,$$

and the *deterministic communication complexity* of c is

$$D(c) := \min_{\mathcal{P}} |\mathcal{P}|,$$

where the minimum is taken over all correct protocols for c.

We need to allow for randomness in the protocols to get lower bounds on randomized streaming algorithms. In a *randomized protocol*, each player i is given a random string r_i, independent of the other players' strings, and each function b_v is computed on the players input and random string. A randomized protocol can be viewed as a probability distribution over deterministic protocols. A randomized protocol Π is correct for c if for all $x \in \mathcal{X}$

$$P_{\mathcal{P} \sim \Pi}(\mathcal{P}(x) = c(x)) \geq 2/3.$$

The *cost* of a randomized protocol Π is

$$|\Pi| := \max_{\mathcal{P} \in \mathrm{supp}(\Pi)} |\mathcal{P}|,$$

and the *randomized communication complexity* of c is

$$R(c) := \min_{\Pi} |\Pi|,$$

where the minimum is taken over all correct randomized protocols for c.

Two additional definitions will also be important. The first is a restriction to *one-way* protocols. A one-way protocol is one in which each player has only one opportunity to write to the shared blackboard and the players write in order, player 1 first, then player 2, etc., with player t last. The one-way communication complexity of c is the minimum cost of a one-way protocol for c. $D_{1-way}(c)$ and $R_{1-way}(c)$ denote the deterministic and randomized one-way communication complexities of c, respectively.

The second is a *promise problem*. Any communication problem can become a promise problem by selecting a subset $\mathcal{L} \subseteq \mathcal{X}$ and "promising" that the input is a member of that subset. The only modification our earlier discussion is a weaker condition for the correctness of protocols. Specifically, any protocol for a promise problem must be correct for inputs from $\mathcal{L}$, but it can behave arbitrarily on inputs from $\mathcal{X} \setminus \mathcal{L}$.

Now we describe three functions that are used in storage lower bounds for streaming algorithms.

2.5.1 Functions

INDEX_s

INDEX was one of the first functions used for streaming lower bounds. Alon, Matias, and Szegedy [3] reduce from INDEX to prove a $\Omega(n)$ lower bound on the space complexity of any algorithm that approximates the maximum frequency in a stream with at most n distinct items. In INDEX_s, there are two players Alice, who is given a subset $A \subsetneq [s]$, and Bob, who is given an index $b \in [s]$, or formally $\mathcal{X}_1 = 2^{[s]}$, $\mathcal{X}_2 = [s]$, and $\text{INDEX}_s(A, b) = 1$ if and only if $b \in A$. The one-way randomized communication complexity of this function is $\Omega(s)$, i.e. $R_{1-way}(\text{INDEX}_s) = \theta(s)$ [75, p. 49].

$\text{DISJ}_{s,t}$

The t-party disjointess problem $\text{DISJ}_{s,t}$ was introduced by Alon, Matias, and Szegedy [3] for lower bounds on the complexity of approximating the frequency moments. The multiparty disjointness problem $\text{DISJ}_{s,t}$ is a promise problem for $t \geq 2$ players. Each player $i = 1, 2, \ldots, t$ is given a set $A_i \subseteq [s]$ such that either

1. $A_i \cap A_j = \emptyset$, for all $i \neq j$, or
2. there is a unique $d \in [s]$ such that $A_i \cap A_j = \{d\}$, for all $i \neq j$.

The problem is for final player to determine whether the present instance satisfies 1 or 2. It is known that $R(\text{DISJ}_{s,t}) = \Omega(s/t)$ [30, 49].

$\text{DISJ+IND}_{s,t}$

We introduce the problem DISJ+IND in order to prove lower bounds on one pass g-sum algorithms when g is nondecreasing. An instance of the problem appears as an instance of $\text{DISJ}_{s,t+1}$ with the additional promise that the set given to the final player, player $t+1$, is a singleton. As the name suggests, the problem has features of DISJ and INDEX. The idea of adding an extra "index" player was introduced by Li and Woodruff [77] in their Augmented L_∞ Promise Problem for a strengthened lower bound on the space complexity of approximating the frequency moments.

Theorem 7. $R_{1-way}(\text{DISJ+IND}_{s,t}) = \Omega(s/t \log s)$.

Proof. We give a reduction from $\text{DISJ}_{s,t+1}$. Let $\mathcal{P}$ be any randomized protocol for $\text{DISJ+IND}_{s,t}$. Run in parallel $\ell = \lceil 96 \log s \rceil$ independent copies of $\mathcal{P}$ through the first t players. This produces ℓ transcripts. Player $t+1$ now takes the each of the transcripts and computes the final value of each once for every element he holds, as if it was the only element he held. No communication is need for this part because $\mathcal{P}$ is a one-way protocol and $t+1$ is the final player.

Player $t+1$ then takes the majority vote among the independent copies of $\mathcal{P}$ for each element. If any vote signals an intersection then he reports intersection; otherwise he reports disjoint.

If every one of the $|A_{t+1}| \leq s$ majorities is correct then the final player's report is correct. Let X_i, for $i \in A_{t+1}$, be the number among the ℓ copies of $\mathcal{P}$ with the correct outcome when the final player completes protocol using $i \in A_{t+1}$. Then X_i is Binomially distributed from ℓ trials with success probability at least $2/3$. Using a Chernoff Bound we find

$$P(X_i \leq \ell/2) = P\left(X_i \leq (1 - \frac{1}{4})\frac{2}{3}\ell\right) \leq \exp\left\{\frac{-1}{32}\mu\right\} \leq \exp\left\{\frac{-1}{32} \cdot \frac{2}{3}\ell\right\} \leq \frac{1}{s^2}.$$

Thus, with probability at least $1 - \frac{1}{s}$ every majority vote is correct, hence our $\text{DISJ}_{s,t}$ protocol is correct for $s \geq 3$.

Let $T_1, T_2, \ldots, T_\ell$ be the transcripts. The total cost of this DISJ protocol is $\sum_{i=1}^{\ell} |T_i|$. Since $\text{DISJ}_{s,t+1}$ requires $\Omega(s/t)$ bits of communication, at least one of the protocols has length $\Omega(s/t\ell) = \Omega(s/t \log s)$, hence $|\mathcal{P}| = \Omega(s/t \log s)$ bits of communication. □

Remark 8. The reduction above leaves a gap between the upper and lower bounds for the communication complexity of DISJ+IND. Which is correct? If $t = 1$ then the upper bound is correct, $\Omega(s/t)$ communication is required because this is INDEX.

The rest of this chapter contains the bulk of our results. In the next three sections we apply the material that we have reviewed up to now to streaming sum problems in three regimes. Section 2.6 discusses general functions g. It demonstrates a few abberant behaviors and uses the SIMPLESKETCH to approximate a few types of functions. These approximations are likely not optimal. We end the section with the example of approximating the geometric mean of the positive values of $|f|$.

Sections 2.7 and 2.8 discuss monotone functions. There our algorithms and lower bounds are nearly optimal; they characterize the complexity in terms of n, ϵ, and m; and the algorithms are universal.

2.6 Arbitrary functions

This section explores two questions. First, what properties of g are necessary in order that g admits a sublinear space approximation algorithm? We will show that in order for there to be any hope of finding a sublinear space $(1 \pm \epsilon)$-approximation algorithm for a fixed function g it must be that g is nonnegative (or nonpositive) and if g has zeros other than at the origin then it must be periodic. Our results for these cases apply when g is fixed, i.e. it does not depend on n.

The next question is: what can be done to approximate an arbitrary function g that has the necessary properties? We describe a simple, but generally not optimal, algorithm for $(1 \pm \epsilon)$-approximations when g is an arbitrary function based on pairwise independent, uniform sampling.

Given access to g, we calculate a sketch size s_g such that $g(f)$ can be approximated by sampling each frequency pairwise independently with probability $s_g/|\operatorname{supp}(f)|$, leading to a sketch of size $\tilde{O}(s_g)$ bits. Aside from simplicity, a sketch of this form has at least one desirable attribute – it follows from Proposition 1 that it is universal for all functions g' with $s_{g'} \leq s_g$.

2.6.1 Lower bounds

Intuitively, $g(f)$ ought to be difficult to approximate when small changes in the frequencies can result in large changes in its value. This is the case when g takes both positive and negative values because $g(f)$ can be near 0 even when m is large. It is also the case when $g(x) = 0$, for some $x > 0$. The next two theorems establish linear lower bounds on any the storage complexity of these types of functions. It is important to note that in this section (though, not in Sections 2.7 and 2.8) the constants in the asymptotic expressions bounding the space complexity can depend on g. Thus, these theorems do not apply if g varies with the size of the stream.

Theorem 9. *If there exist integers $x, y \in \mathbb{N}$ such that $g(x) > 0 > g(y)$, then for sufficiently small, though constant, ϵ the space complexity for a $(1\pm\epsilon)$-approximation algorithm is $\Omega(n)$. The unspecified constant and ϵ depend only on g.*

Proof. Let $x, y \in \mathbb{N}$ be any integers such that $g(x) > 0 > g(y)$. By Lemma 10 there exists $z \in \mathbb{N}$ such that $g(x+z) - g(x) \neq g(z)$ or $g(y+z) - g(y) \neq g(z)$. There is no loss in generality to assume that $g(x+z) - g(x) \neq g(z)$, because otherwise we can replace g with $-g$ as any $(1 \pm \epsilon)$-approximation algorithm for g can be used for a $(1 \pm \epsilon)$-approximation for $-g$. Let $\mathcal{A}$ be a one-pass $(1 \pm \epsilon)$-approximation algorithm. We prove that $\mathcal{A}$ uses $\Omega(n)$ space by reduction from INDEX.

Let

$$n' = \min\left\{ \frac{n-z}{x + y\frac{g(x)}{-g(y)}}, \frac{n}{1 + \frac{g(x)}{-g(y)}} \right\} = \Omega(n).$$

Alice receives a set $A \subsetneq [n']$ and Bob receives an index $b \in [n']$. Let $n_B = \lfloor |A| \frac{g(x)}{-g(y)} \rfloor$, then $C = |A|g(x) + n_B g(y)$ satisfies $0 \leq C \leq -g(y)$. Alice and Bob jointly create a stream and apply $\mathcal{A}$ to approximate $g(f)$. For each $i \in A_i$, Alice adds $(i, 1)$ to the stream x times. Next, Alice runs $\mathcal{A}$ on the stream and transmits the contents of its memory and the value n_2 to Bob. Bob adds $(i, 1)$ to the stream y times for each of each of $i = n'+1, n'+2, \ldots, n'+n_B$ and adds $(b, 1)$ to the stream z times and finishes the computation. By our choice of n', the domain of the stream is $[n' + n_B] \subseteq [n]$ and its length is $x|A| + yn_b + z \leq n$.

There are two possible outcomes: if $b \notin A$ then

$$g(f) = |A|g(x) + n_B g(y) + g(z) = C + g(z),$$

and otherwise

$$g(f) = C - g(x) + g(x+z).$$

For sufficiently small ϵ, the algorithm $\mathcal{A}$ will distinguish between these two cases because $g(x+z) - g(x) \neq g(z)$. Thus, $\mathcal{A}$ inherits the $\Omega(n') = \Omega(n)$ lower bound of $\text{INDEX}_{n'}$. □

Lemma 10. *Let $g : \mathbb{N} \to \mathbb{R}$. If there exist $x, y \in \mathbb{N}$ such that $g(y) < 0 = g(0) < g(x)$, then there exists $z \in \mathbb{N}$ such that $g(x+z) \neq g(x) + g(z)$ or $g(y+z) \neq g(y) + g(z)$.*

Proof. Suppose, for contradiction that $g(x+z) = g(x) + g(z)$ and $g(y+z) = g(y) + g(z)$ for all

$z \in \mathbb{N}$. Then $g(xy) = yg(x) > 0$ and $g(xy) = xg(y) < 0$, which is a contradiction. □

Remark 11. Strictly speaking, Theorem 9 does not rule out sublinear approximations for the case $g(x) > 0 > g(y)$, for $x > 0 > y$ in the turnstile model (when g is not symmetric). However, the same argument works with the result that any such function with a sublinear approximation algorithm in the turnstile model is linear.

Henceforth we assume $g \geq 0$. Of course, if $g(x) = 0$ for $x \neq 0$ then $g(f)$ can be still be 0 for arbitrarily large streams. Theorem 12 shows that such a function must be periodic with period $\min\{x \in \mathbb{N}_{>0} : g(x) = 0\}$ or else any $(1 \pm \epsilon)$-approximation algorithm requires linear storage. Lemma 13 contains the main part of the reduction.

Theorem 12. *If $g(x) = 0$ for $x > 0$, then for sufficiently small ϵ and $(1 \pm \epsilon)$-approximation algorithm requires $\Omega(n)$ bits, unless g is periodic with period $\min\{z > 0 : g(z) = 0\}$. The unspecified constant and ϵ depend only on g.*

Proof. Let $p = \min\{z \geq 1 | g(z) = 0\}$. If g is not periodic with period p then there is some x such that $g(x) \neq g(x + p)$, choose $\epsilon < |g(x) - g(x + p)|/(g(x) + g(x + p))$. When $n \geq 2(x + p)$ we have $m \geq n \geq x + pn/2p$, so that the hypothesis for Lemma 13 is satisfied with $s = n/2p$. □

Lemma 13. *Let $g : \mathbb{N} \to \mathbb{R}$, and let $Z \subseteq \mathbb{N}$, $Z \neq \emptyset$ be the zeros of g. Let $z \in Z$, $x \in \mathbb{N}$, and $s \leq n$. If $m \geq zs + x$ and $|g(x) - g(x + z)| > 2\epsilon(g(x) + g(x + z))$, then any $(1 \pm \epsilon)$-approximation algorithm requires $\Omega(s)$.*

Proof. The proof is again by reduction from INDEX_s. Alice adds z copies of each element in her set $A \subseteq [s]$ to the stream and Bob adds x copies of his index $b \in [s]$. Alice and Bob use the approximation algorithm to jointly compute $\hat{G}$, a $(1 \pm \epsilon)$-approximation to $g(f)$. The true value is either $g(x)$, if $b \notin A$, or $g(x + z)$, otherwise. If $g(x) > g(x + z)$ then Bob can correctly determine whether $b \in A$ if $(1 - \epsilon)g(x) - (1 + \epsilon)g(x + z) > 0$, which is true by assumption. The case $g(x) < g(x + z)$ is the same. Thus, the algorithm inherits an $\Omega(s)$ space lower bound from INDEX_s. □

2.6.2 Sketching

Suppose a function g exhibits none of the pathologies described in Section 2.6.1, can we approximate $g(f)$ with sublinear space? Maybe, the next theorem shows that the space complexity for any positive function cannot be more than the maximum ratio of its values at two points in the range 1 to m. Thus, if a function does not vary greatly we can approximate it in sublinear space.

Theorem 14. *Let g be a positive function and let*

$$s = \frac{\max_{x\in[m]} g(x)}{\min_{x\in[m]} g(x)}.$$

There is a turnstile streaming algorithm that uses $O(\epsilon^{-2} s \log^2 n \log M)$ and outputs a $(1 \pm \epsilon)$-approximation to $g(f)$ with probability at least $5/8$. The algorithm can be implemented in the insertion only model with $O(\epsilon^{-2} s \log M + \log^2 n)$ bits of memory.

Proof. Let $f \in \mathcal{F}$ be the frequency vector of the stream and let $d \in [n]$. By definition

$$\frac{s}{\epsilon^2 |\operatorname{supp}(f)|} \geq \frac{g(f_d)}{\epsilon^2 g(f)},$$

so we can apply Proposition 1 implemented with the SimpleSketch. Along with Theorem 3, a union bound on the error probability of two implies that the sketch is correct and the result is a $(1 \pm \epsilon)$-approximation with probability at least $1 - (1/4) - (1/8) = 5/8$. □

The linear lower bound for functions with zeros, Theorem 12, does not apply to periodic functions for a good reason – they can be approximated in very small space! Indeed following the previous theorem, all that needs to be established is a method from sampling from the support of f mod p, which are the only frequencies that contribute to $g(f)$. SimpleSketch works here as well provided we perform the updates modulo the period.

Corollary 15. *Bounded positive functions and periodic functions with period* $\min\{x \geq 1 : g(x) = 0\}$ *can be approximated in* $O(\epsilon^{-2}\log^2(n)\log(M))$ *space. The constant depends on the function* g*.*

Proof. Using Theorem 14, it is sufficient to show how to sample from $\operatorname{supp}(f \mod p)$. This is easily accomplished with SimpleSketch by performing all arithmetic modulo p. □

Corollary 15 generalizes the earlier work of McGregor, Rudra, and Uurtamo [80] and Indyk [55]. They describe an algorithm for approximating the number of frequencies not divisible by p, which is the same as g-sum approximation when g is the indicator set of those frequencies. Their algorithm uses $O(\epsilon^{-2}\log n)$ bits of space.

2.6.3 Heavy elements

For completeness, we include a heavy elements algorithm for arbitrary functions. It is akin to SimpleSketch.

Theorem 16. *Let* g *be a positive function and let*

$$s = \frac{\max_{x\in[m]} g(x)}{\min_{x\in[m]} g(x)}. \tag{2.4}$$

There is a turnstile ϵ*-heavy elements algorithm for* g *that uses* $O(\epsilon^{-1}s\log n\log M)$ *with probability at last* $2/3$*. The algorithm can be implemented with* $O(\epsilon^{-1}s\log M + \log^2 n)$ *bits of memory in insertion-only model.*

Proof. The algorithm is to use GetCounts to store $\epsilon^{-1}s$ frequencies or determine that $|\operatorname{supp}(S)| > \epsilon^{-1}s$. It is sufficient to prove that any stream S with an ϵ-heavy element has $|\operatorname{supp}(S)| \leq \epsilon^{-1}s$. If i is an ϵ-heavy element, then

$$\max_{x\in[m]} g(x) \geq g(f_i) \geq \epsilon g(f) \geq \epsilon|\operatorname{supp}(f)| \min_{x\in[m]} g(x).$$

The bound follows. □

Corollary 17. *Let g be a nonnegative function with zeros $p\mathbb{Z}$ and let*

$$s \geq \frac{\max\{g(x) : x \in [m] \setminus p\mathbb{Z}\}}{\epsilon \min\{g(x) : x \in [m] \setminus p\mathbb{Z}\}}.$$

There is a turnstile ϵ-heavy elements algorithm that uses $O(\epsilon^{-1} s \log n \log M)$.

2.6.4 The geometric mean

We conclude Section 2.6 with an example. Consider approximating the geometric mean of the positive absolute frequencies, $M_0(S) = \prod |f_i|^{1/L_0}$. A priori, this does not fall into our framework, so the approach is to take a logarithm, find a suitably good approximation to that sum, and then exponentiate. We have $\lg M_0(x) = \frac{1}{L_0} \sum_{f_i>0} \lg f_i$. We find a sufficiently good approximation X to $\lg M_0(x)$ so that e^X is a $(1 \pm \epsilon)$-approximation to $M_0(x)$.

The upshot of this section is that it is possible to $(1 \pm \epsilon)$-approximate $M_0(S)$ within polylogarithmic space. First, a roadblock and a short detour, $\lg(1) = 0$, so Lemma 12 implies a $\Omega(n)$ lower bound on the space used by any algorithm that outputs a $(1 \pm \epsilon)$-approximation to $\lg(f)$. Fortunately, we can accept additive error in the approximation to $\lg M_0(S)$. The strategy is to perturb the function at $x = 1$, so that the lower bound no longer applies and then find an approximation to the perturbed function with a suitably accurate, multiplicative approximation ratio.

Now for the perturbation, we have

$$\lg(x + \delta) - \lg(x) \leq \lg(1 + \delta) \leq \delta$$

and $\lg(1 + \delta) > \delta/2$ for $x \geq 1$ and $\delta \leq 1$. Let $\delta = \tilde{\Omega}(1)$ and let $g(x) = \log(x + \delta)/\log(1 + \delta) = \tilde{O}(1)$. The function g satisfies the hypotheses of Theorem 14 with $s = 2\lg(m)/\delta$.

We request $(1 \pm \epsilon'/3)$-approximations $\hat{G}$ to $g(f)$ and $\hat{L_0}$ to L_0, where $\epsilon' = \frac{\lg(1+\epsilon)}{2 \lg m}$ and $\delta = \frac{\lg(1+\epsilon)}{2(1+\epsilon')}$. With these choices for the parameters $2^{\log(1+\delta)\hat{G}/\hat{F_0}}$ is a $(1 \pm \epsilon)$ approximation to

$M_0(S)$. Indeed, we have the following upper and lower bounds on the error.

$$\begin{aligned}\exp\left\{\log(2)\frac{1+\epsilon'/3}{(1-\epsilon'/3)L_0}\sum_i g(f_i)\log(1+\delta)\right\} &\leq 2^{(1+\epsilon')\delta+(1+\epsilon')\log M_0(S)}\\ &\leq 2^{(1+\epsilon')\delta+\log M_0(S)+\epsilon'\log m}\\ &= (1+\epsilon)M_0(S)\end{aligned}$$

and

$$\begin{aligned}\exp\left\{\log(2)\frac{(1-\epsilon'/3)}{(1+\epsilon'/3)L_0}\sum_i g(f_i)\log(1+\delta)\right\} &\geq 2^{(1-2\epsilon'/3-O((\epsilon')^2))\log M_0(S)}\\ &\geq 2^{(\log M_0(S)-\epsilon'\log m}\\ &\geq M_0(S)\frac{1}{\sqrt{1+\epsilon}}\\ &\geq (1-\epsilon)M_0(S).\end{aligned}$$

Putting it together, we have the following theorem.

Theorem 18. *There is a streaming algorithm that outputs a $(1\pm\epsilon)$-approximation to the Geometric Mean in one pass with $O(\epsilon^{-3}\log^2 n\log^4 m\log M)$ bits of storage. The algorithm can be implemented with $O(\epsilon^{-3}\log^2 m\log M)$ bits of storage in the insertion only model.*

Proof. From Theorem 14, computing $\hat{G}$ can be done in

$$O(s\log^2 n\log M/\delta(\epsilon')^2) = O(\epsilon^3\log n^2\log^3 m\log M)$$

bits of storage, which dominates the approximation of L_0. □

This may not lead to a space optimal approximation, but it does pretty well in terms of quality-for-effort.

2.7 Decreasing functions

Now we will tackle decreasing functions. We begin with lower bounds on the space complexities of approximating $g(f)$. The lower bound is essentially the maximum value $|\operatorname{supp}(f)|$ over the vectors $f \in \mathcal{F}$ that have a ϵ-heavy element. By construction this implies that the algorithm that stores each nonzero frequency exactly is a nearly optimal algorithm for finding ϵ-heavy elements.

Surprisingly, a pairwise independent sample of the roughly same size suffices as a sketch for $g(f)$, so the space complexity of approximating $g(f)$ is completely characterized, except for a small gap. The dependence on $\mathcal{F}$, in particular the relationship of n and m, has a significant impact on the complexity of the problem, which depends delicately on how m scales with n. To our knowledge, this type of close dependence of approximation complexity on $\|f\|_1$ has not been observed before.

2.7.1 Lower bounds

Supposing that $g(x)$ decreases to 0 as $x \to \infty$, a $\Omega(n)$ lower bound on the space complexity of approximating $g(f)$ is always available with a reduction to INDEX. That fact is a consequence of Theorem 19. However, very long streams may be needed for the reduction. Given only the streams in $\mathcal{S}$, those with n or fewer distinct items and L^1-length m or less, we must weaken the lower bound. Our strategy is to parameterize the lower bound reduction in terms of the frequencies f. Optimizing the parameterized bound over $f \in \mathcal{F}$ gives the best possible bound from this reduction—it leaves only a small gap to the space complexity of our algorithms. This section primarily establishes the next theorem, which is our main lower bound theorem for decreasing functions.

Theorem 19. *Let g be a decreasing function and let*

$$\sigma = \sigma(\epsilon, g, \mathcal{F}) = \max\{|\operatorname{supp}(f)| : f \in \mathcal{F},\ g(f) \leq \epsilon^{-1} g(1)\}. \tag{2.5}$$

Then any p-pass streaming g-sum $(1 \pm \epsilon)$-approximation algorithm requires $\Omega(\sigma/p)$ bits of space.

The value σ is essentially the maximum number of distinct items in a stream that contains a ϵ-heavy element for g. The idea is to take a frequency vector $f \in \mathcal{F}$ with $g(f) \leq \epsilon^{-1} g(1)$ and create a stream with the frequency vector similar to f. A reduction from $\text{DISJ}_{|\operatorname{supp}(f)|,2}$ then gives us a $\Omega(|\operatorname{supp}(f)|)$ lower bound, and (2.5) just chooses the best among those lower bounds.

Lets switch and think about algorithms for a moment, suppose a stream has a frequency vector f with a ϵ-heavy element i. Then $g(1) \geq g(f_i) \geq \epsilon g(f)$ which implies, by the definition of σ, that

$$|\operatorname{supp}(f)| \leq \sigma. \tag{2.6}$$

Of course, we compute $g(f)$ in $O(\sigma \log M)$ bits, slightly worse in the turnstile model, by storing a counter for each element of $\operatorname{supp}(f)$, which is optimal for this instance. Section 2.7.3 gives a uniform approach for handling all frequency vectors, not just those with ϵ-heavy elements.

The proof of Theorem 19 is broken up with a two lemmas. The first one is helpful (although it can be avoided) in the reduction from $\text{DISJ}_{s,2}$. It will show up again later when we discuss a fast scheme for computing σ for general functions.

Lemma 20. *Let $y_i \in \mathbb{R}_{\geq 0}$, for $i \in [s]$, and let $v : \mathbb{R} \to \mathbb{R}_{\geq 0}$. If $\sum y_i \leq Y$ and $\sum v(y_i) \leq V$, then there exists i such that $\frac{s}{2} y_i \leq Y$ and $\frac{s}{2} v(y_i) \leq V$.*

Proof. Without loss of generality $y_1 \leq y_2 \leq \cdots \leq y_\sigma$. Let i_j, $j \in [\sigma]$, order the sequence such that $v(y_{i_1}) \leq v(y_{i_2}) \leq \cdots \leq v(y_{i_s})$ and let $I = \{i_j | j \leq \lfloor s/2 \rfloor + 1\}$. By the Pigeon Hole Principle, there exists $i \in I$ such that $i \leq \lfloor s/2 \rfloor + 1$. Thus $\frac{s}{2} y_i \leq \sum_{j=\lfloor s/2 \rfloor + 1}^{s} y_{i_j} \leq Y$ and $\frac{s}{2} v(y_i) \leq \sum_{j=\lfloor s/2 \rfloor + 1}^{s} v(y_j) \leq V$. □

With Lemma 20 in mind, we could alter the definition of (2.5) to restrict the maximization to streams that have all frequencies equal and still get the same order lower bound. That does appreciably affect computing the lower bound (indeed, this is one of the steps in our algorithm to approximate σ), but it makes reasoning about σ messier. For example, in the discussion above we can no longer conclude as (2.6) does that $|\operatorname{supp}(f)| \leq \sigma$, rather we must again invoke Lemma 20.

Invoking Lemma 20 to change the definition of σ means that we also have to invoke it when comparing $|f|$ with σ, which is need for our later proofs. The definition of σ is left as-is to avoid this rigmarole.

Now for the main reduction of this section. We use DISJ rather than INDEX to get lower bounds on multiple pass algorithms.

Lemma 21. *Let g be decreasing and $\epsilon > 0$. Suppose that $f = (1, y, y, \ldots, y, 0, \ldots, 0) \in \mathcal{F}$ and 1 is an ϵ-heavy frequency, then any streaming g-sum $(1\pm\epsilon)$-approximation algorithm requires $\Omega(|\operatorname{supp}(f)|/p)$ bits of storage.*

Proof. Let $s = \lfloor |\operatorname{supp}(f)|/2 \rfloor$ and let $\mathcal{A}$ be an approximation algorithm. The reduction is from DISJ$(s, 2)$ where Alice receives $A \subseteq [s]$ and Bob receives $B \subseteq [s]$ with the promise that $|A \cap B| \leq 1$. For each $i \in A$, Alice puts $(i, 1)$ in the stream once. She then runs $\mathcal{A}$ on her portion of the stream and sends the contents its memory to Bob. For each $i \notin B$, Bob adds y copies of $(i, 1)$ to the stream. Bob runs $\mathcal{A}$ on his portion of the stream and sends the memory back to Alice. She recreates her portion of the stream, advances $\mathcal{A}$, sends the memory to Bob, etc., until each player has acted p times. In addition to the algorithm's memory, on each pass Alice sends at most $\lceil p^{-1} \lg |A| \rceil$ binary digits of $|A|$ so that Bob knows $|A|$ at the end of the protocol.

The stream is a member of $\mathcal{S}$ by construction; let f' be its frequency vector. At the end, Bob finishes computing $\mathcal{A}(f')$. All of the frequencies are y, $y + 1$, or 1. If

$$\mathcal{A}(f') > (1 + \epsilon)[|A|g(y + 1) + (s - |B| - |A|)g(y)],$$

then Bob declares $b \in A$ and otherwise $b \notin A$.

The exact value of $g(f')$ is either

$$V_1 = g(1) + (|A| - 1)g(y + 1) + (s - |B| - |A| + 1)g(y),$$

if $|A \cap B| = 1$, or

$$V_0 = |A|g(y+1) + (s - |B| - |A|)g(y),$$

if the intersection is empty. Indeed, we find

$$V_1 - V_0 \geq g(1) \geq \epsilon(g(1) + 2sg(y)) \geq \epsilon(g(1) + 2V_0)$$

Hence, if $\mathcal{A}(f')$ is a $(1 \pm \epsilon)$-approximation to $g(f')$, then Bob's decision is correct. The protocol with solves $\textsc{disj}(s, 2)$ which requires, in the worst case, $\Omega(s)$ bits of communication including $O(p^{-1} \lg s)$ bits to send $|A|$ and $\Omega(s) = \Omega(|\operatorname{supp}(f)|)$ bits for $(2p-1)$ transmissions of the memory of $\mathcal{A}$. Thus, in the worst case, at least one transmission has size $\Omega(|\operatorname{supp}(f)|/p)$. □

Proof of Theorem 19. Let $f \in \mathcal{F}$ be a maximizer and apply Lemma 20 to the positive elements of f. From this we find that there exists y such that

$$ys' + 1 \leq \|f\|_1 \qquad \text{and} \qquad g(1) \geq \epsilon(s'g(y) + g(1)),$$

for $s' = (\sigma - 1)/2$. Therefore, $f' = (1, y, y, \ldots, y, 0, \ldots, 0) \in \mathcal{F}$ with $\lfloor s' \rfloor$ coordinates equal to y. Applying Lemma 21 to f' implies the desired bound. □

Immediately from the proof of Theorem 19 we get a lower bound on any (α, ϵ)-heavy elements algorithm.

Corollary 22. *Let g be a decreasing function. Any p-pass streaming (α, ϵ)-heavy elements algorithm for g requires $\Omega(\frac{1}{p}\sigma(\alpha, g, \mathcal{F}))$ bits of space.*

Theorem 24 in Section 2.7.3 shows that the bound in Theorem 19 is tight up to poly-logarithmic factors. In concert with the algorithm in the next section, this shows that the space complexity of approximating decreasing functions depends delicately on the length of the stream. That is in contrast with the situation for F_p, $p \geq 2$, where the $\tilde{\theta}(n^{1-2/p})$ space complexity is correct

for all streams with $m = \text{poly}(n)$. This is also a better bound than Theorem 14.

First, we describe how the value of σ can be computed approximately with only $O(\log m)$ evaluations of g.

2.7.2 Computing $\sigma(\epsilon, g, \mathcal{F})$

The value σ turns out to be approximately the streaming space complexity of g, and it is a parameter that will be needed for our algorithm. That means we need a way to compute it for an arbitrary function. One can get by with just evaluating g at $O(\log m)$ points, as we will now show.

Because g is decreasing, the maximum of (2.5) will be achieved by a vector f of length m. Lemma 20 says that we might as well take all of the other frequencies to be equal, so we can find a near maximizer by enumerating. Specifically, let

$$s(y) = \min\left\{\frac{m}{y}, \frac{(1-\epsilon)g(1)}{\epsilon g(y)}\right\}$$

be the maximum bound we can achieve using y as the single frequency. The value of the maximizer is at most twice $\max\{s(y) : (m/n) \le y \le m\}$.

But we do not need to check every $y = 1, 2, \ldots, m$ to get a pretty good maximizer. It suffices to check only values where y is a power of two. Indeed, suppose that y^* maximizes $s(y)$ and let $y^* \le y' \le 2y^*$. We will show that $s(y') \ge s(y^*)/2$, and since there is a power of two between y^* and $2y^*$ this implies that its s value is at least $s(y^*)/2 \ge \sigma/4$.

Since y^* is a maximizer we have $s(y') \le s(y^*)$, and because $y' \ge y^*$ and g is decreasing we have $g(y') \le g(y^*)$. This gives us

$$(1-\epsilon)g(1)/g(y') \ge (1-\epsilon)g(1)/g(y^*) \ge s(y^*).$$

We also have

$$\frac{m}{y'} \geq \frac{m}{2y^*} \geq \frac{1}{2}s(y^*).$$

Combining these two we have $s(y') \geq s(y^*)/2$.

Thus, one can get by with enumerating at most $\lg m$ values to compute the bound in Theorem 19. Take the large of the $\lg m$ values tried and quadruple it to get an upper bound on σ.

2.7.3 Approximation and heavy elements

This section presents the approximation algorithm and the proof that it is correct. The algorithm is random sampling with SimpleSketch. The main theorem is Theorem 24; it gives an upper bound on the storage complexity of any function that approximates $g(f)$. Because the algorithm is SimpleSketch, it is universal for all decreasing functions with the same or smaller space complexity. The gap versus the lower bounds in Theorem 19 is $O(\epsilon^{-1} \log^2 n \log M)$ in the turnstile model and $O(\epsilon^{-1} \log M)$ in the insertion only model.

We need one lemma. It gives us some control on $\sigma(\epsilon, g, \mathcal{F})$ as ϵ varies. For brevity, we use $\sigma_\epsilon = \sigma(\epsilon, g, \mathcal{F})$ when g and $\mathcal{F}$ are clear from the context.

Lemma 23. *If $\alpha < \epsilon$, then $\epsilon(1 + \sigma(\epsilon, g, \mathcal{F})) \geq \alpha\sigma(\alpha, g, \mathcal{F})$.*

Proof. Let $\sigma_\epsilon = \sigma(\epsilon, g, \mathcal{F})$ and define σ_α similarly. Let $f \in \mathcal{F}$ such that $\sigma_\alpha = |\operatorname{supp}(f)|$ and $g(f) \leq \alpha^{-1}g(1)$, without loss of generality the coordinates are ordered such that $f_1 \geq f_2 \geq \cdots \geq f_{\sigma_\alpha} > 0$. Let $s' = \frac{\alpha}{\epsilon}\sigma_\alpha$, and let f' be the vector that takes the first $\lfloor s' \rfloor$ coordinates from f and is 0 thereafter. The choice is made so that $f' \in \mathcal{F}$ and

$$g(f') \leq \frac{\alpha}{\epsilon}g(f) \leq \epsilon^{-1}g(1).$$

Then, by definition of σ_ϵ, we have

$$\sigma_\epsilon \geq |\operatorname{supp}(f')| = \left\lfloor \frac{\alpha}{\epsilon}\sigma_\alpha \right\rfloor \geq \frac{\alpha}{\epsilon}\sigma_\alpha - 1.$$

□

Theorem 24. *There is a turnstile streaming algorithm that outputs a $(1 \pm \epsilon)$-approximation to $g(f)$ with probability at least $5/8$ and that uses $O(\epsilon^{-1}\sigma \log^2(n) \log(M))$ bits of space. It can be implemented in the insertion model with $O(\epsilon^{-1}\sigma \log(M) + \log^2 n)$ bits of space.*

Proof. We claim that using the SimpleSketch of Algorithm 3 to sample with probability proportionally to $\min\{1, 8\epsilon^{-1}(\sigma+1)/|\operatorname{supp}(S)|\}$ gives the desired approximation factor. Indeed, let $f = f(S)$ and $\alpha = g(1)/g(f)$. If $\alpha \geq \epsilon$ then $|\operatorname{supp}(f)| \leq \sigma_\alpha \leq \sigma_\epsilon$, so the algorithm stores every nonzero frequency and computes $g(f)$ exactly.

Suppose that $\alpha < \epsilon$. For all $d \in [n]$, we have

$$\frac{g(f_d)}{g(f)} \leq \frac{g(1)}{g(f)} \leq \alpha \leq \frac{\epsilon(1+\sigma_\epsilon)}{\sigma_\alpha} \leq \frac{\epsilon(1+\sigma_\epsilon)}{|\operatorname{supp}(f)|},$$

where the second inequality comes from Lemma 23 and the third from the definition of σ_α as a maximum. In particular, this implies that

$$\frac{8\epsilon^{-1}(\sigma+1)}{|\operatorname{supp}(f)|} \geq \frac{8g(f_d)}{\epsilon^2 g(f)},$$

so the Archetypal Sketch, by Proposition 1, gives the desired approximation ratio. The claim now follows from the correctness of SimpleSketch proved in Theorem 3, Corollary 4, and a union bound over their failure probabilities. □

This solves the online $(1 \pm \epsilon)$-approximation problem when the stream's frequency vector f is promised to obey $|\operatorname{supp}(f)| \leq n$ and $\|f\|_1 \leq m$ at every update. Theorem 24, in fact, provides

a stronger guarantee. Namely, the $(1 \pm \epsilon)$-approximation guarantee holds at any point in the stream where the promise is kept, even if the promise has been previously broken and restored.

The important consequence of Theorem 19 is that if there is a heavy element in f, then we know that $|\operatorname{supp}(f)| \leq \sigma$. In this case, it is within a logarithmic factor of optimal to store a counter for each element in $\operatorname{supp}(f)$. The only trouble is to do this is to find the values in $\operatorname{supp}(f)$ in a turnstile stream or determine that it is larger than σ. Fortunately, established techniques handle this problem quite easily.

Theorem 25. *There is an ϵ-heavy hitters algorithm for g that uses $O(\sigma \log(n) \log(M))$ bits of space.*

Proof. The algorithm is simply to store a counter for each nonzero frequency, up to σ counters, using GetCounts, Algorithm 2. If i is an ϵ-heavy element in f, then $g(1) \geq g(f_i) \geq \epsilon g(f)$ hence $|\operatorname{supp}(f)| \leq \sigma$, by the definition of σ. Thus, the algorithm is correct because it records exactly the frequency of every element. □

2.7.4 Generalized means

We are now in a position to deploy the (nearly) matching upper and lower bounds on an example. It will nicely illustrate the trade-off between the length of the stream and the space complexity of the approximation.

The harmonic mean of a sequence of positive real numbers $(x_i)_{i=1}^n$ is

$$M_{-1}(x) = \left(n^{-1} \sum x_i^{-1}\right)^{-1}.$$

More generally, let

$$M_p(x) = \left(n^{-1} \sum x_i^p\right)^{1/p},$$

for $p \neq 0$, and $M_0(x) = \prod x_i^{1/n}$, the geometric mean. M_p is called the p^{th} *generalized mean* of the sequence $(x_i)_{i=1}^n$.

Now, suppose we have a stream $S \in \mathcal{S}$ with frequency vector f, how much space is required in order to approximate the p^{th} generalized mean of the positive values of $|f|$?

If $p > 0$, this is the well-studied frequency moments problem. Indeed, using results of [69] and [3, 59] one can approximate $L_0 = k$ and $\sum |f_i|^p$ using $O(\epsilon^{-2} n^{1-2/p} \log(M))$ space, for $p > 2$, or $O(\epsilon^{-2} \log(M))$ space, for $p \leq 2$.

For $p < 0$, the problem boils down to approximating $g(f)$ for the decreasing function $g(x) = x^p$. It is straightforward to turn such an approximation into an approximation for $M_p(S)$.

For this example it is convenient to treat the frequencies and $|\operatorname{supp}(f)|$ as a continuously variable quantity. Doing so will not change the results. The function x^p is convex, so for fixed $s = |\operatorname{supp}(f)|$, $g(f)$ is minimized when all nonzero elements of f are equal. They should also be as large as possible subject to the constraint that $\|f\|_1 \leq m$, hence the frequencies are each m/s.

From the definition, we have

$$\sigma = \max \left\{ s \leq n : s \left(\frac{m}{s} \right)^p \leq \epsilon^{-1} \right\}.$$

So, quite easily, we arrive at

$$\sigma = \left(\frac{m^{-p}}{\epsilon} \right)^{1/(1-p)}$$

or $\sigma = n$, if that is smaller. This gives us the following theorem.

Theorem 26. *Let $p < 0$. Any streaming algorithm that determines a $(1 \pm \epsilon)$-approximation to $M_p(f)$, for $f \in \mathcal{F}$, requires $\Omega(\min\{n, m^{-p/(1-p)} \epsilon^{-1/(1-p)}\})$ bits of space. Such an approximation can be found with $O(m^{-p/(1-p)} \epsilon^{-(2-p)/(1-p)} \log^2 n \log M + \epsilon^{-2} \log M)$ bits in a turnstile stream and $O(m^{-p/(1-p)} \epsilon^{-(2-p)/(1-p)} \log M + \epsilon^{-2})$ bits in an insertion only stream.*

Proof. The lower bound comes from Theorem 19. For the upper bound, apply Theorem 24 with σ as given above except for ϵ replaced by $\epsilon/3$ to get a $(1 \pm \epsilon/3)$-approximation $\hat{G}$ to $g(f)$ where $g(x) = x^p$. Also use an algorithm of Kane, Nelson, and Woodruff [68] for a $(1 \pm \epsilon/3)$-approximation

$\hat{L}$ to $|\operatorname{supp}(f)|$. The desired approximation is $\hat{L}/\hat{G}$. □

In particular, for the harmonic mean and polynomial $m = n^c$ we have $\sigma = \min\{n, \epsilon^{-1/2} n^{c/2}\}$, and it is apparent that the complexity depends delicately on m.

2.8 Increasing functions

Finally, we discuss $g \in \mathcal{I}$, the increasing functions. This section presents two lower bounds for p-pass streaming g-sum approximation algorithms and a 2-pass heavy elements algorithm that matches the lower bound to within a $O(\log^2 n \log M)$ factor. The algorithm uses CountSketch to identify the heavy elements on the first pass and then determines their frequencies exactly on the second pass. The first lower bound is derived by a reduction from DISJ and applies to all $p \geq 1$. The second lower bound is derived by a reduction from DISJ+IND and applies only to $p = 1$ pass, naturally it is a stronger bound than the first.

The development of the bounds and the 2-pass heavy elements algorithm follows the same strategy as for decreasing functions. The strategy has three main steps:

1. prove a per-stream lower bound with a reduction from a communication problem,
2. optimize the bound over the set of possible streams, and
3. use the optimality to prove correctness for an algorithm.

This has the effect of expressing the storage lower bound as the solution to a nonlinear optimization problem, just as with decreasing functions. Fortunately, it is again easy to approximate the optimal objective function value (i.e. the lower bound) to within a constant factor, and this is enough information to implement the heavy elements algorithm. Although, as before the algorithm requires a priori upper bounds on the dimension of f and $\|f\|_1$.

2.8.1 Lower bounds

Our lower bounds come are inspired by the first lower bound for the frequency moments as proved by Alon, Matias, and Szegedy [3]. The p-pass bound is based on a reduction from multiparty disjointness. For one pass algorithms, the bound may be improved by reducing from DISJ+IND, instead. The first lemma in this section presents the parameterized lower reduction. Subsequently, we state the optimized lower bound.

Lemma 27. *Let $x > z \geq y$ be positive integers and let $f = (x, z, y, y \ldots, y, 0, 0, 0) \in \mathcal{F}$. Let $s = |\operatorname{supp}(f)|$, $t = x/y$, $u = \max\{1, (x-z)/y\}$, and $p \geq 1$. If $g(x) - \lceil t \rceil g(y) \geq \epsilon g(f)$, then any p-pass streaming $(1 \pm \epsilon)$-approximation algorithm for $g(S)$ requires at least $\Omega(\frac{s}{t^2 p}) - \lceil p^{-1} \lg s \rceil$ space. If $g(x) - g(z) - \lceil u \rceil g(y) \geq \epsilon g(f)$, then any 1-pass algorithm requires at least $\Omega(\frac{s}{u^2 \lg s}) - \lceil \lg s \rceil$ space.*

Proof. Let $s' = \lfloor (s-2)/3 \rfloor$. We prove the two bounds, respectively, with reductions from $\text{DISJ}_{s',\lceil t \rceil}$ with blackboard communication and $\text{DISJ+IND}_{s',\lceil u \rceil}$ with one-way communication.

Let $A_1, A_2, \ldots, A_t$ be the sets given to the players and suppose $\mathcal{A}$ is a p-pass $(1 \pm \epsilon)$-approximation algorithm for $g(S)$. The players jointly create a stream where each player i adds y copies of j to the stream, for each $j \in A_i$. The players repeat the following p times. In order $1, 2, \ldots, \lceil t \rceil$ each player reads the current memory of $\mathcal{A}$ from the blackboard, advances the computation of $\mathcal{A}$ on his portion of the stream, and writes the updated memory of $\mathcal{A}$ back to the blackboard. The memory of $\mathcal{A}$ is written to the blackboard a total of $tp - 1$ times at different stages of the computation. Additionally, every player i announces $\lceil p^{-1} \lg s \rceil$ bits of $|A_i|$ to the blackboard on each pass, so $|A_i|$, for each $i \in [t]$, is known by every player at the end of the computation.

Let S denote the stream created by the players, let $a = \sum_i |A_i|$, and let $\hat{G}$ denote the approximation to $g(S)$ returned by $\mathcal{A}$. The final player declares declares that there is an intersection if $\mathcal{A}(S) > (1+\epsilon)ag(y)$. Otherwise, he declares no intersection.

By construction S is a valid input for $\mathcal{A}$. The true value of $g(S)$ is either $ag(y)$, if there is

no intersection, or

$$(a - \lceil t \rceil)g(y) + g(\lceil t \rceil y)$$

if there is an intersection.

We next show that the final player's declaration is correct provided that $\mathcal{A}$ returns a $(1 \pm \epsilon)$ to $g(S)$. Indeed

$$\begin{aligned}(1-\epsilon)[(a - \lceil t \rceil)g(y) + g(y\lceil t \rceil)] - (1+\epsilon)ag(y) &\geq (1-\epsilon)g(x) - \lceil t \rceil g(y) - 2\epsilon a g(y) \\ &\geq \epsilon(s - 2a)g(y) \\ &\geq \epsilon s g(y)/3 \\ &> 0,\end{aligned}$$

where the second inequality follows from

$$g(x) - \lceil t \rceil g(y) \geq \epsilon g(f) \geq \epsilon(g(x) + sg(y))$$

and the third inequality from $a \leq s/3$. Thus, the protocol we have described is correct for t-player $\textsc{disj}_{s',\lceil t \rceil}$.

By Section 2.5.1, the total number of bits exchanged is $\Omega(s/t)$. Each transmission includes the memory of $\mathcal{A}$ and $p^{-1} \lg s$ bits of $|A_i|$ and there are a total of $pt - 1$ transmissions. Thus on at least one occasion, transmitting the memory of $\mathcal{A}$ requires at least $\Omega(s/t^2 p) - \lceil p^{-1} \lg s \rceil$ bits.

The reduction for the one-pass bound from $\textsc{disj+ind}_{s',\lceil u \rceil}$. All of the players behave the same except that the final (i.e. $(\lceil u \rceil + 1)$th) player adds z copies of his element in the stream, and each player i transmits $\sum_{j \leq i} |A_j|$ rather than $|A_i|$.

The final player declares that there is an intersection if $\mathcal{A}(S) > (1+\epsilon)(ag(y) + g(z))$. To show the correctness we must only check that the declaration is correct if $\mathcal{A}$ has returned a

$(1 \pm \epsilon)$-approximation. Indeed, notice that $u \leq t$ hence

$$\begin{aligned}(1-\epsilon)(g(\lceil u\rceil y + z) + (a - \lceil u\rceil)g(y)) - (1+\epsilon)(ag(y) + g(z)) &\geq (1-\epsilon)g(x) - \lceil u\rceil g(y) - (1+\epsilon)g(z) - 2\epsilon a g(y)\\ &\geq \epsilon(s-2a)g(y)\\ &\geq \epsilon s g(y)/3\\ &> 0,\end{aligned}$$

where the second inequality follows from the assumption that

$$g(x) - g(z) - \lceil u\rceil g(y) \geq \epsilon g(f) = \epsilon(g(x) + g(z) + sg(y)).$$

By Theorem 7, the total communication is $\Omega(s/u \log s)$. The communication for this protocol uses $\lceil u\rceil$ transmissions and each transmission includes the current memory of $\mathcal{A}$ and no more than $\lg s$ bits for $\sum_{j\leq i} |A_j|$. Thus, $\mathcal{A}$ uses $\Omega(s/u^2 \log s) - \lceil \lg s\rceil$ bits of memory on at least one transmission. □

The next lemma provides an alternative to the statement of Lemma 27. In particular, when ϵ is not too small we can simplify the heaviness hypothesis on x. Proposition 29 shows that the assumption on ϵ results in no loss of generality.

Lemma 28. *In the notation of Lemma 27, if $\epsilon > 1/\sqrt{s}$ but the weaker inequalities $g(x) \geq \epsilon g(f)$ and $g(x) - g(z) \geq \epsilon g(f)$ hold, then up to a constant the same lower bounds hold as in Lemma 27.*

Proof. If $2\lceil t\rceil > \epsilon s \geq \sqrt{s}$ then the bound is trivial, so we may take

$$2\lceil u\rceil \leq 2\lceil t\rceil \leq \epsilon s.$$

We replace the vector f by a similar vector $f' = (x, z, y, y, \ldots, y, 0, \ldots, 0)$ that has only $\lfloor s/2\rfloor$

frequencies equal to y. Then

$$\begin{aligned} g(x) - \lceil t \rceil g(y) &\geq \epsilon g(f) - \lceil t \rceil g(y) \\ &\geq \epsilon(g(x) + g(z) + \frac{s}{2} g(y)) \\ &\geq \epsilon g(f'), \end{aligned}$$

and similarly for $g(x) - \lceil u \rceil g(y)$. Since $|\operatorname{supp}(f')| \geq |\operatorname{supp}(f)|/3$, applying Lemma 27 with f' gives the same order lower bound. □

We can also bound the ϵ dependence directly from existing bounds.

Proposition 29. *If g is increasing and $\epsilon > 1/\sqrt{n}$, then any p-pass algorithm that outputs a $(1 \pm \epsilon)$-approximation to $g(f)$ requires $\Omega(\epsilon^{-2}/p)$ bits of storage.*

Proof. The claim follows from Woodruff's proof [97] for the frequency moments, which is a reduction from the communication complexity of Gap-Hamming Distance, and Chakrabarti and Regev's newer multi-round communication lower bound on the same [31]. Woodruff proved that the space complexity of approximating the frequency moments, $g'(x) = |x|^k$ for any $k \geq 0$, is $\Omega(\epsilon^{-2})$ by reduction to a Gap-Hamming instance of dimension $O(\epsilon^{-2})$. The proof uses a stream where the only frequencies are 0, 1, and 2, so the only relevant values of g' are $g'(0) = 0$, $g'(1) = 1$, and $g'(2) = 2^k$.

We have already assumed that $g(0) = 0$, and we can further assume that $g(1) = 1$ by scaling the function. Suppose that $g(2) > g(1) = 1$. We choose $k = \lg g(2)$ then we have $g(x) = g'(x)$ for $x \in \{1, 2, 3\}$ and the claim follows directly from Woodruff's and the newer multi-pass. If $g(2) = g(1)$ then let $x \in \mathbb{N}$ be the maximum value such that $g(x) = 1$ (if no such x exists then the function is constant equal to 1, the lower bound was proved by Indyk and Woodruff [58]). We can alter Woodruff's proof so the frequencies are 0, x, and $2x$ and the same result holds. □

We have completed the first step of the strategy, and we have a per-stream lower bound. The next step is to optimize the bound over the set of available frequency vectors.

For each $f \in \mathcal{F}$ and $s \leq |\operatorname{supp}(f)|$ let $t(s,f) = \overline{f}_1/\overline{f}_s$. Let

$$(\sigma, f^\sigma) \in \arg\max_{s,f} \left\{ \frac{s}{t(s,f)^2} : f \in \mathcal{F}, g(\overline{f}_1) \geq \epsilon g(f) \right\}, \tag{2.7}$$

and denote $\tau = t(\sigma, f^\sigma)$. We note that the set above is always nonempty when $n \geq 2$. When the value of ϵ may not be clear from the context we write σ_ϵ and τ_ϵ. One can assume, without loss of generality, that f^σ has only two distinct frequencies and $|\operatorname{supp}(f^\sigma)| = \sigma$. As with decreasing functions, we stick to the broader statement above because it simplifies the correctness proofs to come.

For one pass algorithms and $s \leq |\operatorname{supp}(f)|$ we define

$$v(s,f) = \max\left\{1, \frac{\overline{f}_1 - \overline{f}_2}{\overline{f}_s}\right\}.$$

Now let

$$(\upsilon, f^\upsilon) \in \arg\max_{s,f} \left\{ \frac{s}{v(s,f)^2} : f \in \mathcal{F}, g(\overline{f}_1) - g(\overline{f}_2) \geq \epsilon g(f) \right\}$$

and $\varphi = v(\sigma, f^\upsilon)$. The feasible set in this case could be empty. In that case we set the bound to 1. Without loss of generality, it can be assumed that f^υ has only three distinct frequencies. Theorem 30 describes our main increasing functions lower bound.

Theorem 30. *If $\epsilon \geq 1/\sqrt{n}$ then any $(1 \pm \epsilon)$-approximation algorithm for $g(f)$ taking p-passes requires at least $\Omega(\epsilon^{-2} + \sigma/p\tau^2) - \lceil p^{-1} \lg \sigma \rceil$ bits of space. For $p = 1$, any such an algorithm requires at least $\Omega(\epsilon^{-2} + \upsilon/\varphi^2 \lg \upsilon) - \lg \upsilon$ bits.*

Proof. The ϵ^{-2} dependence comes directly from Proposition 29. Let x be the largest frequency in f^σ and let y be the σth largest. Now form the vector $f = (x, y, \ldots, y, 0, \ldots, 0) \in \mathcal{F}$ with σ nonzero elements and apply Lemma 28 to get the p-pass lower bound. The 1-pass bound is established in the same way from f^υ. □

2.8.2 Approximating σ/τ^2 and υ/φ^2

Our 2-pass heavy elements algorithm requires the value σ/τ^2 as a parameter, so we need a way to compute it or at least approximate it. We can approximate σ/τ^2 to within a factor of 16 by enumerating $O(\log^2 m \log n)$ values. It requires evaluating g at only $\lg m$ different points. By design, there is a maximizer of (2.7) from a vector $f^\sigma = (x, y, y, \ldots, y, 0, \ldots, 0)$ such that $\sigma = |\operatorname{supp}(f^\sigma)|$. Thus, there are only three important parameters x, y, and $|\operatorname{supp}(f^\sigma)|$. As was the case with decreasing functions, we will find near optimal values for all of these parameters by enumerating powers of 2. First, if $x \geq \sigma y/2$ then $2 \geq \sigma/\tau \geq \sigma/\tau^2$, then the bound is trivial and 1 is a sufficiently good approximation. Henceforth suppose that $x < \sigma y/2$.

Suppose $x \leq x' < 2x$, $y/2 < y' < y$, and $\frac{\sigma}{4} < s \leq \frac{\sigma}{2}$, and let $f' = (x', y', y', \ldots, y', 0, \ldots, 0)$ with $|\operatorname{supp}(f')| = s$. We will show that knowing f' is enough as the bound it generates is within the claimed factor of 16 from the best bound. Then $x' + sy' \leq x + \sigma y \leq m$, hence $f' \in \mathcal{F}$, and

$$(1-\epsilon)g(x') \geq (1-\epsilon)g(x) \geq \sigma y \geq sy',$$

so x is ϵ-heavy. Therefore $\sigma/\tau^2 \geq s/t(s, f')^2$ because it is a maximizer. On the other hand,

$$t(\sigma, f^\sigma) = \frac{x}{y} \leq \frac{x'}{y'} = t(s, f') \leq 4\frac{x}{y},$$

thus $s/t(s, f')^2 \geq \sigma/16\tau^2$.

The consequence is that we can choose x', y', and s as powers of 2. There are at most $\lg m$ such values for x' and y' and $\lg n$ for s. Thus we can enumerate all $(\lg^2 m)(\lg n)$ possible triples for these values and check whether each corresponding vector f' is a member of $\mathcal{F}$ and has $g(x') \geq \epsilon g(f)$. Among those that do, we take the largest value of $s/t(s, f')^2$. It is no more than 16 times smaller than the true maximum. This scheme requires evaluating g only $\lfloor \lg m \rfloor$ times.

Now we turn to approximating υ/φ^2 by enumeration. The above observations still apply

in that we can, without loss of generally, take $f = (x, z, y, y, \ldots, y, 0, \ldots, 0)$ and 2-approximations to y, $|\operatorname{supp}(f)|$, and $x - z$ are sufficient. Unfortunately, we are not able to avoid testing many values of x because the optimal interval $[z, x]$ may be very small. In particular, the optimal interval $[z, x]$ captures important, but local, information about g. The result is that we can again achieve a 16-approximation to the optimal bound by testing $O(m \log^2 m \log n)$ different frequency vectors. One vector is evaluated for every combination of $x = 1, 2, \ldots, m$, $(x - z), y = 1, 2, 4, \ldots, 2^{\lg m}$, and $s = 1, 2, 4, \ldots, 2^{\lg n}$.

2.8.3 Heavy elements in two passes

In this section we describe a 2-pass heavy elements algorithm that matches the lower bound given in the last section, up to polylogarithmic factors. The algorithm uses CountSketch to identify the heavy elements on the first pass and the computes their frequencies exactly on the second pass.

Here is the idea. Consider the frequency vector $f = (x, y, y, \ldots, y, 0, 0, \ldots, 0)$ used in the proof of the lower bound, in particular in Lemma 27. The 2-pass lower bound that we achieve this frequency vector is roughly $|\operatorname{supp}(f)|/(x/y)^2 \leq \sigma/\tau^2$. Just rearranging that inequality yields

$$x^2 \geq \frac{\tau^2}{\sigma} |\operatorname{supp}(f)| y^2,$$

in other words, x is $\tau^2/2\sigma$-heavy for F_2! We have already discussed finding heavy elements for F_2 with CountSketch, although the analysis this time requires a bit more care than in Section 2.4.1. Theorem 35 has the details.

The main part of the work is show that (almost) the same heaviness relationship holds for ϵ-heavy elements in a general vector f. To do so, we first focus on an individual stream S with a ϵ-heavy element. Let $f = f(S)$ be its frequency vector and suppose f has some ϵ-heavy frequency. In particular, $g(\overline{f}_1) \geq \epsilon g(f)$ because g is increasing. Our current goal is to prove the following lemma.

Lemma 31. *Let $f \in \mathcal{F}$ and $s \in \arg\max_{s'}\{s'/t(s',f)^2\}$. Then*

$$\frac{s}{t(s,f)^2}\overline{f}_1^2 \geq \frac{1}{8\lg n}\sum_{d=2}^{n}\overline{f}_d^2.$$

We can assume, without loss of generality, that $f = \overline{f}$. In particular, f is nonnegative and the frequencies are in decreasing order of magnitude.

Lemma 31 is the main advancement that allows us to find heavy elements for $g(f)$ in nearly optimal space because it reduces the problem of finding heavy elements for g to that of finding heavy elements for F_2, which can be done with CountSketch. Indeed, suppose that f has an ϵ-heavy element, then 1 is a heavy element and $g(f_1) \geq \epsilon g(f)$. Hence, f is a feasible point in the optimization problem of (2.7). This means that $s/t(s,f)^2 \leq \sigma/\tau^2$, so Lemma 31 implies that

$$f_1^2 \geq \frac{\tau^2}{8\sigma\log n}\sum_{i=2}^{n} f_i^2.$$

In other words, the heavy element is $\tilde{O}(\frac{\tau^2}{\sigma})$-heavy for F_2 and so we can use CountSketch to identify it with $\tilde{O}(\sigma/\tau^2)$ space, which is optimal, up to polylogarithmic factors, for any algorithm that identifies ϵ-heavy elements for g. This is phase three of the strategy presented at the beginning of Section 2.8—use the optimality of the lower bound (i.e. as the solution to an optimization problem) to prove correctness. The main expression of the optimality is in Proposition 32.

The proof of the bound in Lemma 31 comes in several stages that are presented next. The goal is to bound $\sum_d f_d^2$ in terms of $sf_s = \frac{s}{t^2}f_1^2$. The next proposition is the simplest thing that one can do, but it is very useful for bounding $\sum_d f_d^2$ in terms of f_s.

Proposition 32. *For any $s' \in \mathbb{N}$ we have*

$$s' f_{s'}^2 \leq s f_s^2. \tag{2.8}$$

Proof. Let $t' = f_1/f_{s'}$. From the definition of s as a maximizer we have $s'/(t')^2 \leq s/t^2$ and thus

$$s' \left(\frac{f_{s'}}{f_1}\right)^2 \leq s \left(\frac{f_s}{f_1}\right)^2.$$

□

The purpose of Proposition 32 is to control the other frequencies in terms of f_s. First, we bound the contribution from the frequencies smaller than f_s.

Lemma 33. *If $a > 0$ then*

$$\sum_{i \geq s+1} f_i^a \leq \begin{cases} \lg(n) s f_s^{\max\{a,2\}}, & a > 0, \\ \frac{1}{1-2^{1-a/2}} s f_s^a, & a > 2. \end{cases}$$

Proof. If $s = |\operatorname{supp}(f)|$ then the sum is 0 and the statements are trivial, so suppose $s < |\operatorname{supp}(f)|$. According to Proposition 32 we have $f_{s2^i} \leq f_s 2^{-i/2}$, for $i \geq 0$. Now,

$$\sum_{i \geq s+1} f_i^a = \sum_{i=0}^{\lg n} \sum_{j=s2^i+1}^{s2^{i+1}} f_j^a \leq \sum_{i=0}^{\lg n} s2^i f_{s2^i}^a \leq \sum_{i=0}^{\lg n} s2^{i(1-a/2)} f_s^a.$$

It follows that for any $a > 0$

$$\sum_{i \geq s+1} f_i^a \leq \sum_{i \geq s+1} f_i^{\max\{a,2\}} \leq \lg(n) s f_s^{\max\{a,2\}},$$

and for $a > 2$

$$\sum_{i \geq s+1} f_i^a \leq \frac{s f_s^a}{1 - 2^{1-a/2}}.$$

□

Now, we bound the contribution from frequencies smaller than f_1 but larger than f_s.

Lemma 34. *For $a > 0$ we have*

$$\sum_{i=2}^{s} f_i^a \le 4 s f_s^a \cdot \begin{cases} \frac{2}{1-2^{a-2}}, & a < 2, \\ \lg t, & a \le 2, \\ \lg(t) t^{a-2}, & a \ge 2, \\ \frac{2^{a-2}}{2^{a-2}-1} t^{a-2}, & a > 2. \end{cases}$$

Proof. Let $N_\alpha = \#\{j : f_s 2^{\alpha-1} < f_j \le f_s 2^\alpha\}$, and notice that $N_\alpha \le s_\alpha$ where

$$s_\alpha = \max\{j : f_j > f_s 2^{\alpha-1}\}.$$

Obviously, $f_{s_\alpha} > f_s 2^{\alpha-1}$, so by applying Proposition 32 we find that $s_\alpha \le s 2^{-2\alpha+2}$.

Let $\lambda = \lg \frac{f_1}{f_s} = \lg t$, we have

$$\begin{aligned} \sum_{j=1}^{s} f_j^a &\le \sum_{i=0}^{\lambda} N_{\lambda-i} (f_s 2^{\lambda-i})^a \\ &\le \sum_{i=0}^{\lambda} s_{\lambda-i} (f_s 2^{\lambda-i})^a \\ &\le 4 \sum_{i=0}^{\lambda} s f_s^a 2^{(\lambda-i)(a-2)}. \end{aligned}$$

The four inequalities in the claim follow easily. □

Lemma 31 follows easily from these lemmas.

Proof of Lemma 31. Apply Lemmas 33 and 34 to find

$$2\frac{s}{t^2} f_1^2 = 2 s f_s^2 \ge \frac{1}{4 \lg n} \sum_{i=2}^{s} f_i^2 + \frac{1}{\lg n} \sum_{i=s+1}^{n} f_i^2 \ge \frac{1}{4 \lg n} \sum_{i=2}^{n} f_i^2.$$

□

It may come as a surprise that Lemma 31 does not explicitly include ϵ. The heaviness

parameter ϵ is baked into the definition of s and t. Indeed, as ϵ decreases the collection of streaming with an ϵ-heavy element increases, thus giving a larger feasible set for the optimization problem defining s and t (as in Equation 2.7).

Finally, we present the main theorem of this section. It describes our 2-pass heavy elements algorithm.

Theorem 35. *There is a two pass $(\epsilon, 0)$-heavy elements algorithm that uses $O(\epsilon^{-1} \log m + \frac{\sigma}{\tau^2} \log^2 n \log m)$ bits of storage.*

Proof. The algorithm uses CountSketch on the first pass to identify the set of heavy elements. On the second pass it records their frequencies exactly.

Let $k = \lceil \epsilon^{-1} \rceil + 32\sigma \log(n)/\tau^2$. On the first pass, construct a CountSketch with accuracy $\epsilon = 1/2$, error probability $\delta = 1/4$, and array width $b = 256k$ in order to approximate the top k elements. Let f be the frequency vector of the stream, let H be the set of k items returned by the CountSketch, and let ℓ be the number of ϵ-heavy elements for g, we claim that H contains all ϵ-heavy elements for g. Indeed, with probability at least $1 - \delta = 3/4$ the CountSketch gives an additive

$$\gamma = 8\sqrt{\frac{1}{b} \sum_{d \geq k+1} \overline{f}_d^2} \leq \frac{1}{4}\sqrt{\frac{\tau^2}{8\sigma^2 \log n} \sum_{d \geq k+1} \overline{f}_d^2}.$$

approximation to the frequency of every item. Let $\hat{f}_d$, for $d \in [n]$, denote the approximate frequency of d. The CountSketch returns the set of k elements with the highest $|\hat{f}|$ values.

Suppose that i is an ϵ-heavy element for g. If $|\hat{f}_i| - |\hat{f}_d| > 0$ for every element d with $|f_d| \leq \overline{f}_{k+1} \leq |f_i|$ then $i \in H$. For this, first notice that

$$|\hat{f}_i| - |\hat{f}_d| \geq |f_i| - |f_d| - 2\gamma \geq \overline{f}_\ell - \overline{f}_{k+1} - 2\gamma.$$

Furthermore, because of the ordering of $\overline{f}$ we have

$$\overline{f}_{k+1} < \sqrt{\frac{1}{k - \lceil \epsilon^{-1} \rceil} \sum_{d > 1/\epsilon} \overline{f}_d^2} \leq \frac{1}{2} \sqrt{\frac{\tau^2}{8\sigma \log n} \sum_{d > \ell} \overline{f}_d^2},$$

where the first inequality follows from an averaging argument over the $n - \epsilon^{-1}$ smallest frequencies and the second by recognizing that $\ell < 1/\epsilon$ by virtue of the ℓth frequency being ϵ-heavy. Hence by Lemma 31 we have

$$\overline{f}_\ell - \overline{f}_{k+1} > \frac{1}{2} \sqrt{\frac{\tau^2}{8\sigma \log n} \sum_{d > \ell} \overline{f}_d^2} \geq 2\gamma.$$

Thus $i \in H$ and every ϵ-heavy element is identified. The storage required for this sketch is $O(b \log n \log m) = O(\frac{ps}{\tau^2} \log^2 n \log m)$.

On the second pass, the algorithm exactly determines the frequency for each item in H and then returns the ϵ^{-1} items with the largest frequencies. Since g is increasing, this set contains all of the ϵ-heavy elements for g. The space required for the second pass is an additional

$$O(k \log m) = O(\epsilon^{-1} \log m + \frac{\sigma}{\tau^2} \log n \log m).$$

Thus the total storage required by the algorithm is $O(\epsilon^{-1} \log m + \frac{\sigma}{\tau^2} \log^2 n \log m)$. □

In order to implement the algorithm described in Theorem 35, one only needs to know σ/τ^2. In fact, it suffices to get an upper bound for σ/τ^2. An enumeration-based scheme for that approximation was described in Section 2.8.2. This observation has an important consequence. Because of the way that the algorithm depends on the particular function g, only through the value of the lower bound, it is also correct for any other increasing function g' with the same or smaller space complexity. Indeed, it is a universal heavy elements algorithm for the class of increasing functions with the same or better space complexity.

2.9 Conclusion

The basic strategy that led us to develop nearly space-optimal algorithms is to use the frequency vector to parameterize a lower bound reduction, optimize the bound over all of the frequency vectors in $\mathcal{F}$, and then use the optimality to prove correctness of an approximation or heavy elements algorithm. We first applied the strategy to decreasing functions, parameterizing the two player DISJ reduction. The end result is a simple, universal sketching algorithm with space complexity separated from the lower bound by a reasonably small gap of $O(\epsilon^{-1} \log n \log M)$. We suspect that both the lower and upper bounds can be improved. To see why, consider L_0 in turnstile model, which is the function $g(x) = 1$, for $x \neq 0$, and $g(0) = 0$. L_0 is known to have space complexity $\theta(\epsilon^{-2} \log n)$ [68], whereas the lower bound that we present is only $\Omega(\epsilon^{-1})$ and our upper bound $O(\epsilon^{-2} \log n \log M)$. Neither of our bounds is tight in this case.

As for increasing functions, there is more to do. Using our strategy we present a lower bound for p pass algorithms, with an improvement when $p = 1$, and a nearly optimal 2-pass heavy elements algorithm. It remains to extend the heavy elements algorithm to an approximation algorithm and to develop 1-pass heavy elements and approximation algorithms. This type of extension has been developed by several authors working on related problems [59, 24].

Finally, in Section 2.6 we made some easy observations regarding arbitrary functions. However, finding space-optimal approximation algorithms for such functions is still an open problem. It is plausible that the strategy we have developed for decreasing and increasing functions—parameterize a lower bound reduction with the frequency vector and optimize over class of frequency vectors—will work for arbitrary functions as well, although it is unclear which communication complexity problem should be the basis of the bound.

Chapter 3

Sampling contingency tables

3.1 Introduction

Given integer sequences $r = (r_i)_{i=1}^m$ and $c = (c_j)_{j=1}^n$, our task is to sample at random, a binary matrix $A \in \{0,1\}^{m \times n}$ with $\sum_j A_{ij} = r_i$ and $\sum_i A_{ij} = c_j$. These matrices are known as *binary contingency tables* to statisticians and *co-occurrence* matrices to ecologists. The sequences r and c are known as the row and column *margins* of the matrices. We use $\Sigma_{r,c}$ to denote the set of binary contingency tables with row margins r and column margins c. With a moment's thought one will recognize that the problem is equivalent to randomly sampling simple bipartite graph with left degrees c and right degrees r. The bi-adjacency matrix of such a graph is a binary contingency table.

The matrices $\Sigma_{r,c}$ play an important role in the statistical analysis of multivariate categorical data. Often, it is natural to check whether a given matrix $M \in \Sigma_{r,c}$ is a significant outlier—with respect to a well-chosen statistic—compared to a uniform element of $\Sigma_{r,c}$. This approach was already used in several settings (see [33] and references therein). The problem is that efficiently computing the null distribution of that well-chosen statistic, in particular computing a p-value, is impossible at

present. That is true even for very simple statistics, there is presently no efficient algorithm known that can compute the marginal distribution of a single entry for any set of row and column margins!

So, for now we are content with Monte-Carlo estimation of p-values. Unfortunately, there is no fast uniform, or even approximately uniform[1], sampling procedure known to sample from $\Sigma_{r,c}$, which limits the applicability of this approach, too. Our goal is to create samplers that exploit properties of the given degree sequences r and c, to obtain efficient sampling procedures for large and interesting sub-classes of contingency tables.

After some historical overview of the problem, the next sections describe two rejection samplers with fast proposal algorithms and a dynamic programming algorithm that takes time and space that is exponential in the maximum column margin. Our second algorithm, the Columns-In-Expectation sampler of Section 3.5 generates samples from a distribution that may be of independent interest as a replacement for the uniform model.

The remainder of this chapter describes two new "hybrid" contingency table sampling algorithms. Each algorithm runs in two phases, first a dynamic programming algorithm is used to sample part of the table and then a rejection sampler proposes a completion completion of the table.

3.1.1 Three examples

Our initial motivation to study this problem came from a question by Carey Priebe:

How do users interact on online forums?

Observing an internet forum, we can form a matrix A with one row for each user of the forum and one column for each thread. The value A_{ut} is set to 1 if user u posts to thread t or 0 otherwise.

The data come from an online cancer discussion forums. There are several instances; Figure 3.1 has plots of the row and column margins taken from the interactions on a breast cancer forum. For this instance, there are $m = 1775$ users who have posted to one or more of the $n = 3015$ threads

[1]There are fast heuristic algorithms, but none offers a provable guarantee.

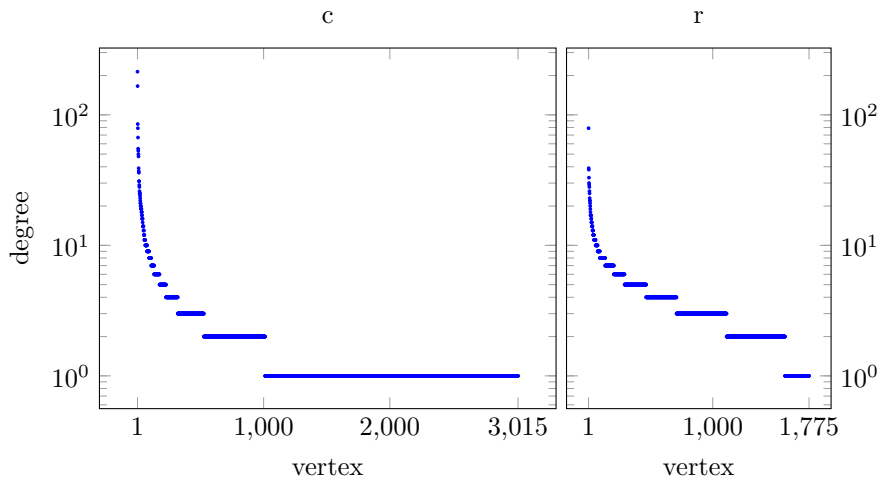

Figure 3.1: The degree sequences c and r for the largest forum instance. There are 3015 threads in a forum with 1775 users.

in the forum. The most active user has posted to $\max_i r_i = 79$ threads and the longest thread has comments from $\max_j c_j = 214$ unique users. However, there are only $\sum r_i = \sum c_j = 6872$ total thread-user pairs, so the resulting graph is very sparse. In fact, the feature that makes this a difficult instance is the few high degree vertices. In the approaches that we present we will find that sparsity and small degrees are helpful features. Although, this instance is still out of reach for our samplers.

The second and third examples come from Jeff Lichtman at Harvard University. Dr. Lichtman is a biologist studying synaptic connectivity in mice. He has experimentally determined the connections between neurons and muscles in several mice, and has the hypothesis that the structure of the connections can be explained within a low-dimensional space. In our setting, we identify a set X containing m neurons and a set Y containing n muscles. In a typical instance, which is the connectivity graph in a single mouse, we have $m = 6$, $n = 133$, and the degree sequences are $r = (45, 37, 35, 34, 31, 31)$ and $c = (3^{(7)}, 2^{(66)}, 1^{(60)})$, where $i^{(j)}$ represents j copies of the integer i. In this instance, the graph is relatively dense, but the maximum degree for a vertex in Y is only 3.

In another experiment, Lichtman has identified a group of 1017 axons and a group of 547 dendritic spines and determined that 7651 of the axon-spine pairs are close enough to form a synapse.

Among those, only 1031 actually do form a synapse. Let T denote the 547×1017 binary "touch" matrix where $T_{ij} = 1$ if spine i and axon j are close enough to synapse and $T_{ij} = 0$, otherwise. The touch matrix has maximum row and column margins of 52 and 10, respectively. Similarly, let S be the 547×1017 binary "synapse" matrix that identifies which pairs do synapse. S has row and column margins

$$r = (1^{(292)}, 2^{(135)}, 3^{(61)}, 4^{(32)}, 5^{(12)}, 6^{(11)}, 7^{(2)}, 8, 10)$$

and $c = (1^{(1003)}, 2^{(14)})$, respectively. The goal is to perform a hypothesis test about the matrix S. As before we will fix the margins, but the matrix T also imposes structure on synapses. Indeed, the desired distribution in this case is uniform on $\{A \in \Sigma_{r,c} : A \leq T\}$.

3.2 Preliminaries

We are given as input vector $r \in \mathbb{N}^m$ and $c \in \mathbb{N}^n$ and our task to sample uniformly at random a binary $m \times n$ matrix with row margins r and column margins c. Let $\Gamma_{r,c} \subseteq \mathbb{N}^{n \times m}$ be the set of all nonnegative integer matrices with row margins r and column margins c and let $\Sigma_r \subseteq \{0,1\}^{m \times n}$ and $\Gamma_r \in \mathbb{N}^{m \times n}$ be the corresponding sets of matrices with row sums r and any column sums.

In light of the third example above, we define the *bounded contingency table* sampling problem. In the bounded contingency table sampling problem, we are given a matrix $B \in \mathbb{N}^{m \times n}$ of cell bounds in addition to the row and column margins. Let $\Gamma^B_{r,c}$ denote the set of all $m \times n$ nonnegative integral matrices A with row margins r, column margins c, and satisfying $A \leq B$ element-wise We define Γ^B_r analogously. The task now is to sample a matrix uniformly from $\Gamma^B_{r,c}$. Clearly, if every entry of B is 1 then $\Sigma_{r,c} = \Gamma^B_{r,c}$, and otherwise the sets may differ. We assume without loss of generality that for all i, j the cell bounds satisfy $B_{ij} \leq \min\{r_i, c_j\}$.

First, it is important to understand when $\Gamma^B_{r,c}$ is non-empty. This problem can be solved efficiently with a network flow computation or a linear program. Here is a network model for the existence problem. Define a bipartite network with vertices $s, t, u_1, u_2, \ldots, u_m, v_1, \ldots, v_n$ and edges

su_i for all i, u_iv_j for all i, j, and v_jt for all j. The capacity of each edge is given in Table 3.1.

Edge type	Capacity
su_i	r_i
u_iv_j	B_{ij}
v_jt	c_j

Table 3.1: Edges and capacities to determine whether $\Gamma^B_{r,c} = \emptyset$ or not.

The maximum flow in this network is $\sum_i r_i$ if and only if $\Gamma^B_{r,c}$ is not empty. An integral maximum flow always exists because the data are integral. Given an integral maximum flow f with value $\sum_i r_i$, the matrix $A \in \mathbb{N}^{m\times n}$ with $A_{ij} = f(u_iv_j)$ is a member of $\Gamma^B_{r,c}$. With the best available max-flow algorithm testing feasibility this way requires $O(|V||E|) = O(nm^2 + n^2m)$ time [72, 85].

For non-emptiness of $\Sigma_{r,c}$ there is a simpler criterion discovered by Gale [45] and Ryser [90]. Let us reorder the row margins so that $r_1 \geq r_2 \geq \cdots \geq r_m$, and let $c^*_k = \#\{j : c_j \geq k\}$ for $k = 1, 2, \ldots, m$.

For $\Sigma_{r,c}$ to be nonempty it is obviously necessary that $c^*_1 \geq r_1$. A bit of additional investigation reveals that it is also necessary that

$$\sum_{j=1}^{k} c^*_j \geq \sum_{j=1}^{k} r_j,$$

for $k = 1, 2, \ldots, m$, with equality when $k = m$. What Gale and Ryser each proved is that these m inequalities are sufficient as well.

Treating c as a partition of the integer $\sum_j c_j$, for a moment, the vector c^* is its dual partition in the sense that the Ferrer's Diagram of c^* is the transpose of the Ferrer's Diagram of c. When c^* and r are related in this way one says that c^* *majorizes* r. Majorization is a well studied concept in mathematics and has important ramifications in linear algebra and the theory of inequalities [79]. The Gale-Ryser criterion can be stated succinctly as: the Ferrer's dual of the column margins majorizes the row margins.

If we could calculate $|\Gamma^B_{r,c}|$ for any choice of r, c, and B then we could calculate the necessary

marginal distributions in order to sample a table one element at a time. However, if B is a square, binary matrix and $r = c = \mathbf{1}$ then $|\Gamma^B_{r,c}|$ is equal to the permanent of B, which was famously shown to be #P-complete[2] by Valiant [94]. And the difficulty does not just come from the bounds as Dyer, Kannan, and Mount have shown that computing $|\Gamma_{r,c}|$ is also #P-complete [42]—even when $m = 2$. Perhaps surprisingly, the complexity of counting the matrices in $\Sigma_{r,c}$ is unknown.

Some notation will simplify the exposition. The following will be used throughout this chapter. Let $u, v \in \mathbb{N}^n$ be integral vectors of the same dimension n, we write $v! = \prod_{i=1}^n v_i!$, $u^v = \prod_{i=1}^n u_i^{v_i}$, and $\binom{u}{v} = \prod_{i=1}^n \binom{u_i}{v_i}$.

3.3 Samplers galore

One might, naïvely, try a rejection algorithm that assigns the outcomes of mn independent Bernoulli random variables to the entries of the table and accepts the resulting matrix if the margins are correct. If the Bernoulli random variables are identically distributed then it is obvious that this algorithm gives perfectly uniform samples. The problem with this approach is, of course, that the potential samples are drawn from a set of size 2^{mn}, which is typically exponentially larger than $|\Sigma_{r,c}|$.

There are two typical routes to such an algorithm. The first is to refine the set on which the rejections take place. The second is to allow the samples to be drawn from a slightly non-uniform distribution.

All existing methods have serious drawbacks. They generally fall into two categories. Either the algorithms are too slow to practically sample from even moderately sized instances or actual sampling distribution is unknown. However, there are important classes of margins for which satisfactory solutions exist, most notably when the column margins are bounded by a small constant and when the margins are all equal.

Jerrum and Sinclair [61] were able to reduce the problem of sampling from $\Sigma_{r,c}$ to that of

[2]This means that if there is a polynomial time algorithm for computing the permanent then there is a polynomial time algorithm that counts the number of satisfying assignments to a Boolean formula, one consequence of which would be P=NP.

sampling a random perfect matching in a bipartite graph. A perfect matching in an n vertex bipartite graph can be sampled approximately uniformly with a Markov chain that mixes in $O(n^6)$ steps [60], but the reduction involved increases the size of the problem quadratically, which is unappealing in light of the already slow sampling time. Bezáková, Bhatnagar, and Vigoda [17] describe a Markov chain that mixes in $O(n^8)$ steps (for a $n \times n$ matrix), which is faster but still impractical for even moderate instances like those described earlier.

Diaconis and Gangolli [39] introduced a symmetric Markov chain on $\Gamma^B_{r,c}$. The transition rule is to choose a random 2×2 submatrix and attempt a random "swap" on that submatrix. A swap is either (1) add 1 to each diagonal and subtract one from each off-diagonal or (2) do the reverse, each with probability 1/2. If the new matrix is a member of $\Gamma^B_{r,c}$ then the swap is completed, or if not then the chain holds. Diaconis and Gangolli prove that the chain is irreducible over $\Gamma_{r,c}$ and it is known to be rapidly mixing for $\Gamma_{r,c}$ in many instances [40, 34].

When the bounds are included, the picture is less clear. Unfortunately, the chain is known to have exponential mixing when B is a general binary matrix [16] (even when the chain is irreducible). On the other hand, Kannan, Tetali, and Vempala [70] show that it is rapidly mixing for instances describing regular graphs. The mixing time for $B = 1$, i.e. when the state space is $\Sigma_{r,c}$, is an open problem.

Chen, Diaconis, and Holmes [33] introduced a popular sequential importance sampler that is available as a package for the R statistical computing environment [1]. The algorithm samples one column at a time from a simple proposal distribution. Although the algorithm is fast to produce matrices there is no guarantee on their quality. In fact, the sampling distribution is known to be highly non-uniform in some instances [18].

Harrison and Miller [52] modify the proposal distribution of the Chen-Diaconis-Holmes importance sampler. The new proposal sampling algorithm incorporates some dynamic programming to better approximate the true marginal distribution of the to-be-proposed column. Their algorithm also runs quickly but provides no guarantee on the sampling distribution.

Still more algorithms are known for sampling from $\Gamma^B_{r,c}$, they typically apply to a particular range of the parameters. In the upcoming sections we discuss two additional samplers from the literature and some new samplers. Those from the literature are the well-studied Configuration Model (Section 3.4) and a dynamic programming algorithm (Section 3.6).

3.4 The Configuration Model

We find it simpler to describe this sampler in terms of bipartite graphs (and their bi-adjacency matrices) rather than contingency tables. The Configuration Model is a probability distribution on bipartite multigraphs with given degrees, or equivalently on $\Gamma_{r,c}$. We will soon see that sampling from the Configuration Model is simple and the time required to generate one sample is linear in the number of edges. Furthermore, conditionally given that a randomly sampled graph is simple its distribution is the uniform distribution on $\Sigma_{r,c}$. This makes the Configuration Model a promising candidate for a rejection sampling algorithm. The downside is that the acceptance probability is very small unless the resulting graphs are very sparse. However, sparsity was a feature of our first example—the one with its margins shown in Figure 3.1. Experimentally, the Configuration Model rejection sampler does not succeed on this instance, but it does work for another one of the cancer forum instances.

The Configuration Model was introduced by Bollobás as a theoretical tool for counting labelled regular graphs [21]. Wormald was the first to suggest it as an algorithmic tool for sampling graphs [98].

3.4.1 Sampling

Algorithm 4 samples a contingency table from the Configuration Model. It is simple to describe in terms of bipartite graphs. The vertices will be $X = \{u_i\}_{i=1}^m$ and $Y = \{v_j\}_{j=1}^n$. First, we assign r_i distinct "handles" to each vertex $u_i \in X$ and c_j distinct handles to each vertex $v_j \in Y$. Next while

Algorithm 4 Sample a contingency table from the Configuration Model

```
procedure ConfigurationModelSampler(r, c)
    A_ij ← 0, for i ∈ [m] and j ∈ [n]
    c' ← c
    for i = 1, 2, ..., m do
        for ℓ = 1, 2, ..., r_i do
            Sample j ∈ [n] proportionally to c'_j
            A_ij ← X_ij + 1
            c'_j ← c'_j − 1
        end for
    end for
    return A
end procedure
```

any X handle remains, we choose a X handle and then choose a Y handle uniformly at random (and independently of the other choices), add an edge between the corresponding vertices to the graph, and remove the handles. The resulting multigraph is a sample from the Configuration Model. One easily finds that an equivalent description of the model is to choose a random bijection between the handles of X and Y.

To sample from $\Sigma_{r,c}$ we add a rejection step as well. If the resulting graph is simple then we accept the sample. Otherwise, we reject and perform another, independent trial. Effectively, the algorithm just samples a random matching between the X and Y handles and simplifies the matching to a multigraph. Any other method for doing so will work as well.

It is easy to see that the preceding rejection algorithm always returns a bipartite graph with the correct degree sequence (presuming that one exists) and that it can generate any such graph. To prove that the sampling distribution is uniform is not much harder.

Theorem 36. *Let X be drawn from the Configuration Model with Algorithm 4. Then $X \in \Gamma_{r,c}$ and for all $A \in \Gamma_{r,c}$ we have*

$$P(X = A) = \frac{r!c!}{(\sum_i r_i)!} \prod_{ij} \frac{1}{A_{ij}!}.$$

In particular, the distribution of X conditionally given that it is binary is the uniform distribution on $\Sigma_{r,c}$.

Proof. Let $A \in \Gamma_{r,c}$ and let G be the corresponding bipartite multigraph. We give each edge of G two labels. First, we label the edges of G as $1, 2, \ldots, N = \sum r_i$ arbitrarily.

Let $\mathcal{G}$ be the multigraph returned by choosing a random bijection between the handles on the left and the handles on the right. The graph $\mathcal{G}$ has the same distribution as the output of Algorithm 4, thus it suffices to prove the theorem for $\mathcal{G}$. On the other hand, we are free to define the process generating the random bijection. To choose the bijection we repeatedly select one unmatched handle in each set and match them with an edge, and as we proceed, we label the edges of $\mathcal{G}$ as $1, 2, \ldots, N$ in the order that they are added to $\mathcal{G}$.

The probability that $\mathcal{G} = G$ and both of the labellings match is

$$\frac{\left(\prod_{i=1}^{m} r_i!\right)\left(\prod_{j=1}^{n} c_j!\right)}{N!N!}.$$

The probability that $\mathcal{G} = G$ is the sum of the probabilities over all distinct labellings of G. There are $N! \prod_{ij} \frac{1}{A_{ij}!}$ distinct labellings of G because permuting the labels among the edges joining any single pair of vertices does not change the labelling. Thus,

$$P(\mathcal{G} = G) = \left(\frac{N!}{\prod_{ij} A_{ij}!}\right) \frac{r!c!}{N!N!} = \frac{r!c!}{N! \prod_{ij} A_{ij}!}, \tag{3.1}$$

as desired. Notice that (3.1) does not depend on the organization of A, but only on the multiset of distinct values in A. In particular, the probability is the same for all $A \in \Sigma_{r,c}$ and equal to $r!c!/N!$. □

Unfortunately, this rejection sampler is typically not efficient. For example, the following theorem shows that just two vertices of relatively high degree is enough to ruin it. This is reason why it doesn't work for the breast cancer forum instance in Figure 3.1.

Theorem 37. *Consider a sequence of margin pairs* $(r^{(N)}, c^{(N)}) \in \mathbb{N}^{m(N)} \times \mathbb{N}^{n(N)}$ *with* $\sum r_i^{(N)} =$

$\sum c_j^{(N)} = N$. *Let* $d_r^{(N)} := \max_i r_i$, $d_c^{(N)} := \max_j c_j$. *If*

$$\lambda = \lambda(N) := \frac{d_r^{(N)} d_c^{(N)}}{N} = \omega(\log N),$$

then the expected running time of the handle sampling algorithm is super-polynomial in N.

Proof. Let $u \in X$ and $v \in Y$ be vertices of degree d_r and d_c, respectively, and let Z be the number of times the edge $\{u, v\}$ appears in the candidate graph. We have

$$P(Z = 0) = \frac{(N - d_c)!/(N - d_c - d_r)!}{N!/(N - d_r)!} \leq \left(1 - \frac{d_c}{N}\right)^{d_r} = \left(1 - \frac{\lambda}{d_r}\right)^{d_r} \leq e^{-\lambda}$$

and

$$P(Z = 1) = \frac{d_r d_c (N - d_c)_{d_r - 1}}{(N)_{d_r}} \leq \lambda N P(Z = 0).$$

Hence,

$$P(Z \leq 1) \leq (1 + \lambda N) e^{-\lambda},$$

which is super-polynomial in N since $\lambda = \omega(\log N)$. Of course, the event $\{Z \leq 1\}$ contains the event that the candidate graph is simple, this establishes the super-polynomial expected running time. □

A much stronger result has been proved by Blanchet and Stauffer. It includes the following necessary condition.

Theorem 38 (Blanchet & Stauffer [20])**.** *Let* $X^{(N)}$ *be a sample from the Configuration Model with margins* $r^{(N)}$ *and* $c^{(N)}$. *If* $P(X^{(N)} \in \Sigma_{r^{(N)}, c^{(N)}}) = \Omega(1)$ *as* $N = \sum r_i = \sum c_j \to \infty$ *then*

$$\sum_{i,j} r_i^{(N)} (r_i^{(N)} - 1) c_j^{(N)} (c_j^{(N)} - 1) = O(N^2).$$

3.4.2 Sampling proportionally to a weight function

It will be helpful to conceptualize the Configuration Model sampler as an algorithm that iteratively samples edges proportionally to a changing weight function. The weights to which we refer are the number of unused handles at each vertex.

Consider organizing Algorithm 4 to sample one entire row at time, each with the correct marginal distribution conditionally given what has already been fixed. One finds from Theorem 36 that we should sample the first row $A_1 \in \mathbb{N}^n$ with probability proportional to $\prod_{j=1}^{n} \frac{1}{A_{1j}!}(c_j)_{A_{1j}}$. Then update the column sums as $c_j \leftarrow c_j - A_1$, and repeat with the next row. In general, we sample the ith row $A_i \in \mathbb{N}^n$ with probability proportional to

$$\prod_{j=1}^{n} \frac{1}{A_{ij}!} \left(c_j - \sum_{\ell=1}^{i-1} A_{\ell j} \right)_{A_{ij}}$$

The total probability to sample a given matrix A is proportional to

$$\prod_{j=1}^{n} \frac{c_j!}{\prod_{i=1}^{m} A_{ij}!}.$$

In fact, we have just proved a slightly stronger statement of Theorem 36. Given any multiset V of mn integers let $\Gamma_{r,c}|_V \subseteq \Gamma_{r,c}$ be the set of contingency tables that have as entries the values V. The Configuration Model is uniform over $\Gamma_{r,c}|_V$, as long as the latter is nonempty.

3.5 The Columns-In-Expectation Sampler

This section presents a new cell-bounded contingency table sampling algorithm that produces random tables obeying the bounds and with the correct row sums, but the tables only match the column sums in expectation. It comes in two parts. The first component, algorithmically, computes a set of weights $w \in \mathbb{R}^n_{>0}$. The weights are used in a row-by-row procedure that samples each row inde-

pendently and are chosen so that the sampler gives the correct column sums in expectation. The second component is the sampling procedure that samples from matrices from Γ_r^B with independent rows. This is the only one of the samplers presented in this chapter that incorporates cell bounds. It is always efficient at sampling binary tables, but tables typically do not have the correct column margins.

Let $w \in \mathbb{R}_+^n$ be a set of positive weights and let $B \in \mathbb{N}^{m\times n}$. We will sample a matrix $A \in \Gamma_r^B$ proportionally to

$$w(A) := \prod_{j=1}^{n} w_j^{\sum_i A_{ij}},$$

for a well chosen set of weights. We denote the corresponding probability distribution on $\mathbb{N}^{m\times n}$ as $CiE(w)$.

The next section describes the sampling procedure and Section 3.5.2 describes the initialization of the weights.

3.5.1 Sampling

The sampling is accomplished one row at at time, so it suffices to begin with the case $m = 1$. Given $a, b \in \mathbb{N}^n$ let $w(a) = \prod_{i=1}^n w_i^{a_i}$ we have

$$\Gamma_r^b = \{a \in \mathbb{N}^n : \|a\|_1 = r \text{ and } a \le b\}.$$

Given $1 \le r \le n-1$ our goal is to sample a vector $a \in \Gamma_r^b$ proportionally to $w(a)$. This problem is easily solved with a dynamic program.

We define a table $T \in \mathbb{R}^{(r+1)\times(n+1)}$ where

$$T_{i,j} = \sum \left\{ w(a) : a \in \mathbb{N}^j, \|a\|_1 = i, a_k \le b_k \text{ for } k \le j \right\}, \tag{3.2}$$

for $i = 0, 1, \ldots, r$ and $j = 0, 1, \ldots, n$. In particular $T_{0,0} = 1$ and

$$T_{r,n} = \sum_{a \in \Gamma_r^b} w(a).$$

For convenience, we define $T_{i,j} = 0$ when i or j is negative.

Once T is known, the sampling can be accomplished with Algorithm 5.

Algorithm 5 Sample one row of a matrix subject proportional to a set of weights

procedure SAMPLEROW(T, r)
  Select $U_1, U_2, \ldots, U_n \overset{iid}{\sim} \text{Uniform}(0, 1)$
  $i \leftarrow r$
  for $j = n, n-1, \ldots, 1$ **do**
    $X_j \leftarrow \min\{\ell : U_j T_{i,j} \leq \sum_{k=0}^{\ell} w^k T_{i-k,j-1}\}$
    $i \leftarrow i - X_j$
  end for
  return X
end procedure

It remains to show how the table can be computed efficiently and to prove that the sampling distribution is the desired one. First, initialize $T_{0,0} = 1$ and $T_{i,0} = 0$, for $1 \leq i \leq r$. Every integral vector a of length j and $\|a\|_1 = i$ can be broken into its last coordinate $a_j \in \{0, 1, \ldots, b_j\}$ and a length $j-1$ vector that sums to $i - a_j$. This yields the recursion

$$T_{i,j} = \sum_{k=0}^{b_j} w_j^k T_{i-k,j-1}, \tag{3.3}$$

for $1 \leq j \leq n$ and $0 \leq i \leq r$. The time required to compute the table with the recursion (3.3) is $O(r \sum b_i)$.

The next proposition shows that the sampling distribution is correct for one row.

Proposition 39. *Let $a \in \Gamma_r^b$ and let X be the random vector returned by Algorithm 5. We have*

$$P(X = a) = \frac{w(a)}{\sum_{a' \in \Gamma_r^b} w(a')}.$$

Proof. Let $s_i = \sum_{j=i}^{n} a_i$. A quick application of Equation (3.3) yields

$$P(X = a) = \frac{w_n^{a_n} T_{r-s_n,n-1}}{T_{r,n}} \cdot \frac{w_{n-1}^{a_{n-1}} T_{r-s_{n-1},n-2}}{T_{r-s_n,n-1}} \cdots \frac{w_1^{a_1} T_{0,0}}{T_{r-s_2,2}} = \frac{w(a)}{T_{r,n}}$$

□

Now for a matrix with more than one row. Given bounds $B \in \mathbb{N}^{m \times n} = (b_1, b_2, \ldots, b_m)$, we can sample a table with row sums $r_1, r_2, \ldots, r_m$ as follows. First, initialize by computing a table $T^{(i)}$ for each row i of the final matrix. Second, sample each row independently using Algorithm 5. Notice that if the bounds matrix B has all identical rows, then one only needs to compute the table T for the maximum row margin. Then the table for any other row i is simply the rows zero through r_i of T.

Algorithm 5 guarantees that row i has margin r_i for every $1 \leq i \leq m$. However, the column sums are random.

Given an $m \times n$ matrix A and weights $w \in \mathbb{R}_+^n$ let $w(A) = \prod_{i,j} w_j^{A_{i,j}}$. If c is the vector of column margins of A then $w(A) = w(c)$.

Theorem 40. *Let a matrix X be chosen randomly by sampling each row independently with Algorithm 5 and let $A \in \Gamma_r^B$. Then*

$$P(X = A) = \frac{w(A)}{\sum_{y \in \Gamma_r^B} w(y)}.$$

In particular, the distribution of X conditionally given that its column sums are $c \in \mathbb{N}^n$ is uniform on $\Gamma_{r,c}^B$.

Proof. Let c be the column sums of A and let $A_1, A_2, \ldots, A_m$ be the rows of A. According to Proposition 39,

$$P(X = A) = \prod_{i=1}^{m} \frac{w(A_i)}{T_{r_i,n}^{(i)}} = \frac{w(c)}{\prod_i T_{r_i,n}^{(i)}},$$

which proves both claims. □

3.5.2 Existence and uniqueness of the weights and initialization

The goal of this section is to give necessary and sufficient conditions on r, c, and B such that there exists unique weights $w \in \mathbb{R}^n_{>0}$ so that a matrix drawn from $CiE(w)$ has column margins c in expectation.

A necessary condition and some assumptions

We begin with some assumptions on r, c, and B. We will assume that $\Gamma^B_{r,c} \neq \emptyset$, i.e. there is some nonnegative integral matrix satisfying the bounds with the correct row and column margins. Next, without loss of generality, we may assume that $r, c > 0$, $r_i < \sum_j B_{ij}$, for all $i \in [m]$, and $c_j < \sum_i B_{ij}$, for all $i \in [n]$. Otherwise we may reduce the size of instance by removing the (fixed) row or column.

Given the cell bounds B we define a graph G_B as follows. The vertices of G_B are the column indices $[n]$ and there is an edge $j \sim k$ if and only if there exists a row i so that $B_{ij} > 0$ and $B_{ik} > 0$. Thus, we could rearrange B to be block diagonal so that each block corresponds to a component of G_B. Clearly, the problem of finding weights to satisfy the expectation criterion can be solved independently on each connected component of G_B. Henceforth, we assume that G_B has only one component.

Theorem 41. *Assume that $r_i < \sum_j B_{ij}$, for all $i \in [m]$, and that G_B has a single component. Let w be a set of positive weights and let $C_1, C_2, \ldots, C_n$ be the (random) column sums of a matrix drawn from $CiE(w)$. If $EC_j = c_j$, for all $j \in [n]$, then we have*

$$\sum_{j \in S} c_j < \sum_{i=1}^{n} \min\{r_i, \sum_{j \in S} B_{ij}\}, \text{ for every } \emptyset \neq S \subsetneq [n]. \tag{3.4}$$

Proof. Fix a set $S \subseteq [n]$ and notice that

$$\max_{A \in \Gamma^B_r} \sum_{i=1}^{n} \sum_{j \in S} A_{ij} = \sum_{i=1}^{n} \min\{r_i, \sum_{j \in S} B_{ij}\},$$

where "$\leq$" holds for every $A \in \Gamma_r^B$ and "$\geq$" holds by making a greedy allocation in to columns of S in each row of A. Recall that the support of $CiE(w)$ is all of Γ_r^B. Thus, in order to demonstrate (3.4) is suffices to prove the existence of a matrix A^* with

$$\sum_{i=1}^{n}\sum_{j\in S} A^*_{ij} \leq \max_{A\in\Gamma_r^B} \sum_{i=1}^{n}\sum_{j\in S} A_{ij}.$$

Let A be a maximizer of the right side above. We will find $i \in [m]$, $j \in S$, and $j' \in [n] \setminus S$ that allow us to transform A by moving one unit from the ij entry to the ij' entry and still satisfy the bounds everywhere. The connectedness of G_B implies that there a row i such that $R = \{j \in [n] : B_{ij} > 0\}$ has nonempty intersection with S and with $[n] \setminus S$. Let $j \in R \cap S$ be a column with $A_{ij} \geq 1$. At least one such column exists because $r_i, \sum_{j\in S} B_{ij} > 0$. Let $j' \in R \cap ([n] \setminus S)$ be a column with $A_{ij'} < B_{ij'}$. Such a column exists, for if $\min\{r_i, \sum_{j\in S} B_{ij}\} = r_i$ then any element of $R \cap ([n] \setminus S)$ will do, and if the bounds are the minimizer then $r_i < \sum_{j=1}^{n} B_{ij}$ implies its existence. The new matrix A^* is a copy of A, except the following two entries: $A^*_{ij} = A_{ij} - 1$ and $A^*_{ij'} = A_{ij'} + 1$. By construction $A^* \in \Gamma_r^B$, which completes the proof. □

Before we proceed with the weights, let us handle the case

$$\sum_{j\in S} c_j = \sum_i \min\{r_i, \sum_{j\in S} B_{ij}\}, \text{ for some } \emptyset \neq S \subsetneq [n]. \tag{3.5}$$

Our assumption that $\Gamma_{r,c}^B \neq \emptyset$ implies that (3.4) holds with the "less than" replaced by a "less than or equal to", so we do not need to worry about "greater than". It turns out that we can test whether (3.5) holds, and if it does then the problem can be reduced.

If (3.5) holds then for every row i

$$\sum_{j\in S} A_{ij} = \min\{r_i, \sum_{j\in S} B_{ij}\}, \text{ for all } A \in \Gamma_{r,c}^B.$$

Furthermore, for some row i^* it must be that $\min\{r_{i^*}, \sum_{j\in S} B_{i^*j}\} = \sum_{j\in S} B_{i^*j}$, otherwise $\sum_i r_i =$

$\sum_{j \in S} c_j$ which implies that $c_j = 0$, for all $j \notin S$. Obviously, columns with zero margins can be removed. Since the sum is a minimizer, it means that problem becomes infeasible if any of the bounds $B_{i^* j}$, $j \in S$, is reduced. This least to a simple algorithm for testing whether the necessary condition holds. We just one-at-a-time reduce each nonzero bound by 1 unit and test feasibility with the flow problem described in Section 3.2. If we ever find the problem infeasible then we reduce the corresponding row and column margins by B_{ij} and set B_{ij} to 0. The updated problem is equivalent to the original one, and making such an update at one location in the table does not affect the others. Thus, it suffices to check the feasibility once for each pair i, j with $B_{ij} > 0$. This can be done in $O(n^2 m^3 + n^3 m^2)$ with the best available max-flow algorithm. Henceforth, we assume that (3.4) holds, which also implies that G_B is connected.

The right weights

The marginal probability that the entry in row i and column j of a random matrix generated with the procedure above is equal to k is

$$p_{ijk} = \left(\sum_{\substack{a \in \Gamma_{r_i}^{B_i} \\ a_j = k}} w(a) \right) \Bigg/ \left(\sum_{a \in \Gamma_{r_i}^{B_i}} w(a) \right),$$

for $0 \leq k \leq B_{i,j}$, where B_i denotes the ith row of B. Thus, the expected column margins of such a random matrix A are $E \sum_i A_{i,j} = \sum_{i,k} k p_{ijk}$ and can be computed efficiently with a recursion similar to (3.3).

Our goal is to choose the weights so that $\sum_{i,k} k p_{ijk} = c_j$, for all $1 \leq j \leq n$. This is a nonlinear root finding problem and under some mild assumptions it turns out that there exists a unique positive set of weights that gives the desired columns margins in expectation.

Clearly, scaling all of the weights equally does not change the distribution. So we can pick a particular column and assume without loss of generality that the weight for that column is 1. Next we will assume that $c_j < \sum_i B_{ij}$, for all j. This is also without loss of generality. Indeed, if

$c_j > \sum_i B_{ij}$ then there is no distribution on matrices cell-bounded by B that has jth column mean c_j (in particular $\Gamma^B_{r,c} = \emptyset$). If $c_j = \sum_i B_{ij}$ then the problem can be reduced by one column since the jth column of any matrix with entries bounded by B must be the same jth column as B. Similarly, we will assume that for every pair $i \in [m]$ and $j \in [n]$ there exists $A \in \Gamma^B_r$ with $A_{ij} = 0$, which is equivalent to $\sum_j B_{ij} - r_i \geq \max_j B_{ij}$, for all $i \in [m]$. If this does not hold then we can reduce the instance by subtracting $\min_{A \in \Gamma^B_r} A_{ij}$ from r_i, c_j, and B_{ij}, and then proceeding with the reduced instance only to add the value back to the final matrix.

We prove existence by describing a convergent sequence of weights where the corresponding column expectations converge to c, which is sufficient because the column expectations are a continuous function of the weights. Truncating the sequence once the column expectations are close to the desired values yields an Algorithm 6, which is one way to approximately compute the weights in practice. A root-finding algorithm, like Newton's Method, could be used as well.

The existence and uniqueness of the desired weights is proved in Theorem 46. We need several lemmas for the proof.

Algorithm 6 Compute sampling weights for the Columns-In-Expectation sampler.

```
procedure SetWeights(r,c,ε)
    w ← 1
    while There exists j such that |E_w C_j − c_j| > ε do
        Select j ∈ arg max{c_j − E_w C_j}
        Increase w_j until E_w C_j = c_j
    end while
    return w
end procedure
```

Lemma 42. *Suppose that $E_w C_j < c_j < \sum_i B_{ij}$. Then there exists $\alpha \in (w_j, \infty)$ such that $E_{w'} C_j = c_j$, where w' is the weight vector that has w_j replaced by α.*

Proof. Let $A \sim CiE(w')$. It is sufficient to show that $\lim_{\alpha \to \infty} P_{w'}(A_{ij} = B_{ij}) = 1$, for all $1 \leq i \leq m$, since this implies that $E_{w'} C_j \to \sum_i B_{ij} > c_j$, which implies the claim because $E_{w'} C_j$ is a continuous function of α. Indeed, the convergence is trivial if $B_{ij} = 0$, so suppose $B_{ij} > 0$. By assumption

$B_{ij} \leq r_i$, hence there exists $a \in \Gamma_{r_i}^{B_i}$ with $a_j = B_{ij}$. It follows from direct computation that

$$P(A_{ij} = B_{ij}) = \left(\sum_{\substack{a \in \Gamma_{r_i}^{B_i} \\ a_j = B_{ij}}} \alpha^{B_{ij}} \prod_{i \neq j} w_i^{a_i} \right) \Bigg/ \left(\sum_{\substack{a \in \Gamma_{r_i}^{B_i} \\ a_j = B_{ij}}} \alpha^{B_{ij}} \prod_{i \neq j} w_i^{a_i} + \sum_{\substack{a \in \Gamma_{r_i}^{B_i} \\ a_j < B_{ij}}} \alpha^{a_j} \prod_{i \neq j} w_i^{a_i} \right) \to 1, \quad (3.6)$$

as $\alpha \to \infty$. □

Lemma 43. *Let $A_k = \{j \in [n] : E_w C_j \leq c_j\}$ be the set of indices with lower than desired expected margins after k iterations of Algorithm 6. The sequence of sets A_k is nondecreasing.*

Proof. Let $j \in A_{k-1}$. If j is selected for increase in step j, then trivially $j \in A_k$. Suppose that the column $i \neq j$ is selected for increase. A short calculation shows that

$$\frac{\partial}{\partial w_i} E_w C_j = EC_j \frac{C_i}{w_i} - EC_j E \frac{C_i}{w_i} = \frac{1}{w_i} Cov(C_j, C_i) \leq 0,$$

hence $j \in A_k$. □

Lemma 44. *Let w be a set of nonnegative weights and $E_w C_j < c_j$, for some j. Let u be the corresponding weights where the jth coordinate has been increased so that $E_u C_j = c_j$, but the other coordinates are unchanged. Then*

$$u_j \geq w_j \left(2 - \frac{E_w C_j}{c_j} \right)^{1/N},$$

where $N = \max_j \sum_i B_{ij}$.

Proof. We calculate

$$\begin{aligned}
c_j - E_w C_j &= E_u C_j - E_w C_j \\
&= \frac{\sum_a a_j u_j^{a_j} \prod_{i \neq j} w_i^{a_i}}{\sum_a u_j^{a_j} \prod_{i \neq j} w_i^{a_i}} - \frac{\sum_a a_j w_j^{a_j} \prod_{i \neq j} w_i^{a_i}}{\sum_a w_j^{a_j} \prod_{i \neq j} w_i^{a_i}} \\
&\leq \frac{\sum_a a_j (u_j^{a_j} - w_j^{a_j}) \prod_{i \neq j} w_i^{a_i}}{\sum_a w_j^{a_j} \prod_{i \neq j} w_i^{a_i}} \\
&= \sum_a \left[\left(\frac{u_j}{w_j} \right)^{a_j} - 1 \right] a_j \frac{w(a)}{\sum_{a'} w(a')} \\
&\leq \left[\left(\frac{u_j}{w_j} \right)^{N} - 1 \right] \sum_a a_j \frac{w(a)}{\sum_{a'} w(a')} \\
&\leq c_j \left[\left(\frac{u_j}{w_j} \right)^{N} - 1 \right],
\end{aligned}$$

where the first two inequalities follow because $u_j \geq w_j$. The lemma follows by rearranging. □

Lemma 45. *Suppose that* (3.4) *holds. Let* $w, w' > 0$ *be weight vectors such that there exists a nonempty set* $S \subsetneq [n]$ *for which* $w_j > w'_j$*, for all* $j \in S$*, and* $w_j = w'_j$*, for all* $j \notin S$*. Then* $E_w C_S > E_{w'} C_S$*.*

Proof. Let A be a randomly sampled matrix and denote by A_{iS} the sum of the S entries in the ith row. If $P(A_{iS} = \min\{r_i, \sum_{j \in S} B_{ij}\}) < 1$ and there exist $j \in S$ and $j' \notin S$ such that $B_{ij} > 0$ and $B_{ij'} > 0$ then direct calculation shows $E_w A_{iS} > E_{w'} A_{iS}$; if there are no such j and j' then $E_w A_{iS} = E_{w'} A_{iS}$. Hence $E_w A_{iS} \geq E_{w'} A_{iS}$. The inequality is strict for at least one row i because of the assumption (3.4). Finally, the lemma follows by noting $C_S = \sum_i A_{iS}$. □

Theorem 46. *Suppose that* (3.4) *holds. There exists a unique set of weights* $w \geq 1$ *such with at least one weight equal to* 1 *and* $E_w C_j = c_j$ *for all* j*.*

Proof. For existence, consider Algorithm 6, but remove the termination criterion.

Recall $A_k = \{j \in [n] : E_{w^{(k)}} C_j \leq c_j\}$, where $w^{(k)}$ is the set of weights held by the algorithm at the start of the kth iteration. If $A_k = [n]$, for some k, then $E_{w^{(k)}} C_j = c_j$, for all $j \in [n]$, because $\sum_j C_j = \sum_j c_j$. Thus, a set of weights providing the desired expectations exists.

Now suppose that $A_k \subsetneq [n]$, for every iteration k. We will first prove that the weights $w^{(k)}$ are bounded above. By Lemma 43 there exists a column j^* such that $c_{j^*} < E_{w^{(k)}} C_{j^*}$, for every iteration k, hence $w_{j^*}^{(k)} = 1$ always. Suppose for contradiction that some subset $S \subseteq [n]$ of the columns have unbounded weights. Because w_{j^*} stays at 1 we know $S \neq [n]$. A calculation like (3.6) now shows that

$$P\left(\sum_{j\in S} C_j = \sum_i \min\{r_i, \sum_{j\in S} B_{ij}\}\right) \to 1.$$

On the other hand, Lemma 43 implies that $S \subseteq A_k$ for all sufficiently large k. In particular, by (3.4)

$$\sum_{j\in S} C_j \leq \sum_{j\in S} c_j < \sum_i \min\{r_i, \sum_{j\in S} B_{ij}\},$$

a contradiction. Thus, the sequence of weights is bounded.

Next we show that for any $\epsilon > 0$, there is k_ϵ such that $|c_j - E_{w^{(k)}} C_j| \leq \epsilon$ for all $k \geq k_\epsilon$. Let k such that $|c_j - E_{w^{(k)}} C_j| > \epsilon$. Because $\sum_j C_j = \sum_j c_j$ and there are only n columns, it must be that $c_j - E_{w^{(k)}} C_j \geq \epsilon/n$ for some j and any maximizer of that expression, in particular. Let j be the column that the algorithm chooses for a weight increase. Lemma 44 implies that

$$w_j^{(k+1)} \geq w_j^{(k)} \left(2 - \frac{E_{w^{(k)}} C_j}{c_j}\right)^{1/N} \geq w_j^{(k)} \left(1 + \frac{\epsilon}{n}\right)^{1/N}.$$

Thus, $w_j^{(k+1)} \geq w_j^{(k)}(1+\epsilon')$ for some $\epsilon' > 0$ that depends only on ϵ. Since the weights are bounded, this increase can only happen a finite number of times, and for all iterations k' thereafter $\max_j |c_j - E_{w_j^{(k')}} C_j| < \epsilon$. This completes the proof of existence.

Now for uniqueness. Suppose, for contradiction, that there are two weights $w \neq w'$ such that $E_w C_j = E_{w'} C_j = c_j$, for all j. Let $S = \{j : w_j > w'_j\}$ and $C_S = \sum_{j\in S} C_j$. Without loss of generality, $S \neq \emptyset$. Two applications of Lemma 45 yield the following contradiction

$$\sum_{j\in S} c_j = E_w C_S > E_{w''} C_S > E_{w'} C_S = \sum_{j\in S} c_j,$$

where w'' is the coordinate-wise maximum of w and w'. □

The correctness of Algorithm 6 follows as a corollary.

Corollary 47. *If $c_j < \sum_i B_{ij}$ for every $1 \leq j \leq n$, then Algorithm 6 terminates and the final weights satisfy $|c_j - E_w C_j| \leq \epsilon$, for all j.*

Remark 48. In practice, there are at least two options to find the weights. One can use Algorithm 6, but then we recommend modifying the weight increase step so that the new expected column margin exceeds the old one by exactly ϵ. This will quicken the progress, without the risk of overshooting the correct weight by more than the tolerance.

Second, one can use a root-finding algorithm like Newton's Method. This may be preferable since robust root-finding libraries are freely available.

3.5.3 Nonequivalence with Barvinok's maximum entropy sampler

Barvinok [12] has studied the maximum entropy distribution on $\{0,1\}^{m \times n}$ with independent coordinates, expected row margins r, and expected column margins c. The maximum entropy distribution shares some characteristics with the Columns-In-Expectation distribution, which we now discuss. In addition to the expectations guarantee, the maximum entropy distribution conditioned that the outcome is binary is the uniform distribution on $\Sigma_{r,c}$. The Bernoulli parameters can be found in polynomial time by solving a convex optimization problem, so one could use the maximum entropy model as a surrogate for uniform sampling from $\Sigma_{r,c}$.

Barvinok uses the distribution for bounds on $|\Sigma_{r,c}|$ and $|\Gamma^B_{r,c}|$, when B is binary, and he proves that when the entries in $\frac{1}{n}r$ and $\frac{1}{m}c$ are bounded away from 0 and 1 the sum of any sufficiently large set of entries in a uniformly random matrix from $\Sigma_{r,c}$ concentrates near the expectation of the same sum in the maximum entropy model.

The maximum entropy distribution is characterized by a $m \times n$ matrix of Bernoulli parameters p_{ij}. The marginal distribution of the ijth coordinate in a random matrix from this model

is Bernoulli(p_{ij}) and all of the coordinates are independent. The parameters maximize the entropy $-\sum_{ij} p_{ij} \log p_{ij}$ subject to the constraints $\sum_i p_{ij} = c_j$ and $\sum_j p_{ij} = r_i$.

One can ask whether the maximum entropy distribution conditioned so that the row sums are correct also gives the columns-in-expectation guarantee. The answer is no, conditioning changes the column means.

The coordinates of the maximum entropy distribution are independent, so conditioning on the row sums is the same as sampling one row at a time with Algorithm 5. It can be shown that the weights we should use for the ith row are $w_{ij} = p_{ij}/(1 - p_{ij})$, for $j = 1, 2, \ldots, n$. Generally, these weights do not guarantee the columns of a randomly chosen matrix have the correct margins in expectation. This is true even if the weights are the same for every row, as in the Columns-In-Expectation sampler.

Here is a concrete example. Take column margins $c = (2, 1)$ and the row margins $r = (1, 1, 1)$. First, we compute the maximum entropy matrix and then we show that the conditioned column expectations are not c. Given the matrix of parameters $p = (p_{ij})$, the entropy is the sum of the entropies of its columns. Dropping the row constraints and independently maximizing the entropy of each column gives an upper bound on the maximum attainable entropy. For each column, the maximum is uniquely attained by the constant vector with values $c_j/3$. Thus, every choice for the parameters p gives an entropy no larger than does the matrix

$$p = \frac{1}{3} \begin{pmatrix} 2 & 1 \\ 2 & 1 \\ 2 & 1 \end{pmatrix}.$$

But, this matrix also satisfies the row constraints, so it defines the (in this case unique) maximum entropy distribution.

Now consider the model that samples each entry independently with its corresponding probability p but conditioned so that each row sum is 1. One can easily calculate that the marginal

probabilities for this distribution are

$$p' = \frac{1}{5}\begin{pmatrix} 4 & 1 \\ 4 & 1 \\ 4 & 1 \end{pmatrix},$$

thus the expected column sums are $\frac{1}{5}(12, 3) \neq (2, 1)$.

3.6 Dynamic programming

Another procedure for uniform sampling from $\Sigma_{r,c}$ was devised recently by Miller and Harrison [82]. The algorithm counts the elements of $\Sigma_{r,c}$ by assigning one row and recursing. The recursion tree is enough information to sample uniformly from $\Sigma_{r,c}$ by iteratively computing the marginal distribution of the next row conditionally given the outcome of the rows above it.

A pair of observations explains how Miller and Harrison's algorithm is much more efficient than a naïve recursive enumeration. Given a $\ell \times n$ submatrix A, one of the values in the recursion tree is the number of matrices in $\Sigma_{r,c}$ with A as the initial ℓ rows. Let us, temporarily, denote this value $W(A)$. The important point is that $W(A)$ is independent of the particular arrangement of 1s and 0s in A and depends only on the "unused" portion of the column margins. That is, if we set $r' = (r_{\ell+1}, r_{\ell+2}, \ldots, r_m)$ and $c' = c - \sum_{i=1}^{\ell} A_i$ then the number of completions of A is $|\Sigma_{r',c'}|$, so it is sufficient to have a table entry for each possible vector c' rather than each possible matrix A. The first observation is that although there are many possible choices for A there are usually markedly fewer for c'. The second observation is that the number of entries can be further reduced by exploiting the fact that $|\Sigma_{r',c'}|$ does not depend on the order of the values in c'. The algorithm captures this efficiency by choosing a canonical order for the vectors c'. Let $D = \max_j c_j$. When D is small the second observation provides a lot of traction because the total number of possibilities for c' is no more than n^{d+1} after accounting for different orderings of the same vector (there are n

columns and each has margin 0, 1,..., or d). That reduces the table size to no more than mn^{D+1}.

The time complexity of Miller and Harrison's algorithm is given by the following theorem. The values manipulated by the dynamic program may become very large, so their analysis includes the time required to add or multiply two integers.

Theorem 49 (Miller and Harrison [82]). *There is an algorithm that requires $O(mn^{2D-1}\log^3 n)$ operations to compute $|\Sigma_{r,c}|$ exactly when D is bounded. After this counting step, the algorithm requires $O(mN\log^3 N)$ time for each sample.*

The key part of this result is that D is bounded so their algorithm runs in polynomial time. This is the case for the second example described in Section 3.1.1. There the column margins are necessarily bounded because $m = 6$, although the maximum column margin is only 3.

Let $V(c) \in \mathbb{N}^D$ be the vector with $V(c)_i = \#\{j \in [n] : c_j = i\}$, for $i = 1, \ldots, D$. Given such a vector $v \in \mathbb{N}^D$ let $C(v) = (0^{(n-\sum v_i)}, 1^{(v_1)}, \ldots, D^{(v_D)})$, where $x^{(k)}$ denotes k repetitions of the integer x. Let $r|_i = (r_{i+1}, r_{i+2}, \ldots, r_n)$, for $i = 0, 1, \ldots, n$, and $A|_i$ denote the matrix consisting of rows $i+1, i+2, \ldots, m$ of A. Given $v, a \in \mathbb{N}^D$ we write

$$v \setminus a = (v_1 - a_1 + a_2,\ v_2 - a_2 + a_3, \ldots,\ v_{D-1} - a_{D-1} + a_D,\ v_D - a_D).$$

The vector $v \setminus a$ arises in the following way. Consider an instance with row margins r, column margins $c = C(v)$, and a matrix $A \in \Sigma_{r,c}$. If the first row of A has a_1 1s in columns with margin 1, a_2 1s in columns with margin 2, etc., then, the column margins of the $A|_2$ are $C(v \setminus a)$, neglecting their order.

3.6.1 Sampling

Miller and Harrison's algorithm computes the table

$$W(i, v) = |\Sigma_{r|_i, C(v)}|,$$

for $v \in \mathbb{N}^D$ and $1 \leq i \leq m$, with the convention that $W(m, (0, \ldots, 0)) = 1$.

The following recursion arises when we assign one row of the matrix

$$W(i, v) = \sum_{a \in \mathbb{N}^D} \binom{v}{a} W(i+1, v \setminus a), \tag{3.7}$$

for any $i = 0, 1, \ldots, m-1$ and $v \in \mathbb{N}^D$ such that $W(i, v) > 0$. Indeed, suppose that $\Sigma_{r|_i, C(v)} \neq \emptyset$, hence $W(i, v) > 0$, we are given $a \in \mathbb{N}^D$, and consider building a matrix in $\Sigma_{r|_i, C(v)}$. We begin with the first row by choosing to place a_i 1s in columns with margin v_i, for each $i = 1, 2, \ldots, D$. Each of these choices can be made in $\binom{v_i}{a_i}$ ways. After assigning the first row, the column margins for the remainder of the matrix are described by $v \setminus a$, i.e. the margins are some permutation of $C(v \setminus a)$. The number of ways to configure the remaining $m - i - 1$ rows is $W(i+1, v \setminus a)$. Thus, the vector a has led us to $\binom{v}{a} W(i+1, v \setminus a)$ matrices. Each choice of a leads us to entirely different matrices, owing to a difference in the first row, thus $\sum_{a \in \mathbb{N}^D} \binom{v}{a} W(i+1, v \setminus a)$ is the number if matrices with row margins $r|_i$ and column margins $C(v)$. To be thorough, (3.7) is *not* true when $W(i, v) = 0$, but we can test whether this holds by checking feasibility of the margins $r|_i, C(v)$.

As in the previous section, the recursion (3.7) implicitly defines the marginal distributions that we need in order to sample a matrix from $\Sigma_{r,c}$. Given the table W, the sampling is simple. It is enough to know how to sample a single row. Suppose we have already the first $0 \leq i < m$ rows completed and we want to add the $(i+1)$th row. Let $c' \in \mathbb{N}^n$ be the column margins needed for the remaining $m - i$ rows in order that the completed matrix is a member of $\Sigma_{r,c}$. We sample a vector $a \in \mathbb{N}^D$ with probability

$$\frac{\binom{V(c')}{a} W(i+1, V(c') \setminus a)}{W(i, V(c'))}.$$

To complete the description of the row $i+1$ it remains to place the 1s in the row. There are $\binom{V(c')}{a}$ ways to do so, each is equally likely if the sampling distribution is to be uniform. Therefore we choose, for each $k = 1, 2, \ldots, D$, a random subset of a_k out of the v_k columns in $\{j \in [n] : c'_j = k\}$ and assign a 1 to those columns on row $(i+1)$. The remaining entries on the row are set to 0. Algorithm 7

Algorithm 7 Sample a matrix uniformly with Miller and Harrison's algorithm [82].

procedure MHSAMPLE(r,c,W)
  $X \leftarrow 0 \in \mathbb{R}^{m \times n}$
  for $i = 0, 1, 2, \ldots, m$ **do**
    Select $a \in \mathbb{N}^D$ with probability $\binom{V(c)}{a} W(i+1, V(c) \setminus a)/W(i, V(c))$
    for $k = 1, 2, \ldots, D$ **do**
      Select a a_k-element subset $S \subseteq \{j : c_j = k\}$ at random
      $X_{ij} \leftarrow 1$, for $j \in S$
      $c_j \leftarrow c_j - 1$, for $j \in S$
    end for
  end for
  return X
end procedure

formalizes the procedure. In order to achieve the sampling time claimed by Theorem 49 one must make a judicious choice of data structure. Namely, one must preprocess the table and for each i, v one creates a binary tree containing the positive terms on the right hand side of (3.7). That allows for one to sample each row with only $O(\log n^{D+1}) = O(D \log n)$ operations.

3.6.2 Initialization

Computing the table W is straightforward with the recursion (3.7). Running time is $O(mn^{2D-1} \log^3 n)$ time, $O(mn^D \log^3 n)$ if both r, c bounded, for initialization,

In order to generate the set of possible column margins $v \setminus a$ one can simply test the Gale-Ryser criteria [90] for every vector in $[n]^D$. This can be inefficient if the actual number of feasible vectors is small, but it doesn't affect the running time bound.

Alternatively, we can generate the set more efficiently using a recursive algorithm. The set we want is exactly the set of (unordered) vectors c' so that $r|_i^*$ majorizes c'. We can generate $V(c')$ for each c' in this set, one element at a time from right to left. Here we describe the first step of the recursion.

We begin by determining the maximum and minimum values for $V(c')$. Suppose $D \geq 2$, the case $D = 1$ being trivial. Let x represent the Dth coordinate in a vector $V(c')$. The Gale-Ryser

majorization criteria implies that $Dx \leq \sum_{k=1}^{x} r_k^*$ so

$$\ell = \max\left\{x : Dx \leq \sum_{k=1}^{x} r_k^*\right\}$$

is an upper bound on x (recall that r_k^* decreases with k, so $Dx \leq \sum_{k=1}^{x} r_k^*$ for all $x \leq \ell$). It is easy to check that $x = \ell$ is possible by adjusting the value of the first coordinate.

Now a lower bound on x. After assigning x as the Dth coordinate, there are $n - x$ columns left to assign. In total, the sum of their margins will be $N - Dx$ and each margin is at most $D - 1$. Hence $(n - x)(D - 1) \geq N - Dx$, so by rearranging we get that $k = N - n(D - 1)$ is a lower bound on x. Indeed, $x = k$ can be achieved as well as any value in the interval (k, ℓ).

Now that the range of the Dth coordinate is established, for each value $x \in \{k, k+1, \ldots, \ell\}$ we concatenate x at the end of every vector generated recursively with the replaced values: $n \leftarrow n - x$, $N \leftarrow N - Dx$, $r^* \leftarrow r^*|_x$, and $D \leftarrow D - 1$. The time required for this recursion is D times the length of the set it generates.

Algorithm 8 The dynamic programming algorithm of Miller and Harrison [82].

```
procedure MHInitialize(r,c)
    W(m, (0, 0, ..., 0)) ← 1
    for i = m − 1, m − 2, ..., 0 do
        for v ∈ ℕ^D such that Σ_{r|_i, C(v)} ≠ do
            W(i, v) ← 0
            for a ∈ ℕ^D such that W(i + 1, v \ a) > 0 do
                W(i, v) ← W(i, v) + (v choose a) W(i + 1, v \ a)
            end for
        end for
    end for
    return W
end procedure
```

3.7 A hybrid algorithm

Our last algorithm for sampling uniformly from $\Sigma_{r,c}$ combines elements of Miller and Harrison's dynamic programming sampler and the Configuration Model rejection sampler. The goal is an

algorithm that extends the range of both of those procedures. In particular, the algorithm is targeted for the case when the matrix is sparse with some large row and column margins. The the large margins mean that the Configuration Model rejection sampler fails due to the presence of non-binary entries in the proposals, c.f. Theorem 37, and the initialization time for Miller and Harrison's algorithm is impractically large. The algorithm will make a binary assignment to the top p rows of the table with dynamic programming and then fill in the remaining $m - p$ rows with a sample from the Configuration Model. When the entire resulting table is binary, it will be a uniformly random sample from $\Sigma_{r,c}$.

The algorithm relies on the interpretation, in Section 3.4.2, of the Configuration Model as a weighted sampler. The first step is to use a dynamic program to sample the first p rows of the table A proportionally to

$$u(A) = \frac{c!}{(c - \sum_{i=1}^{p} A_i)!} = \prod_{j=1}^{n} \frac{c_j!}{(c_j - \sum_{i=1}^{p} A_{ij})!}. \tag{3.8}$$

For the remaining $m - p$ rows, we are left with *random* column margins that are the result of the initial sampling procedure. We next draw a proposal from the Configuration Model for this random $(m - p) \times n$ instance. If the proposal is binary it turns out that the combined $m \times n$ table is a uniformly random sample from $\Sigma_{r,c}$. The effect is that the algorithm samples from the Configuration Model conditionally given that the top p rows have only binary entries.

We think of the weight function u as follows. The first 1 in column j of A contributes c_j to the product $u(A)$. The second 1 in that column contributes $(c_j - 1)$ to the product, and so on, so that the ℓth one contributes a factor of $(c_j - \ell + 1)$ to the product $u(A)$. Applying this to the entire matrix, the row $i \leq p$ of A contributes

$$\prod_{j:A_{ij}=1} \left(c_j - \sum_{\ell=1}^{i-1} A_{\ell j} \right) = \prod_{j=1}^{n} \left(c_j - \sum_{\ell=1}^{i-1} A_{\ell j} \right)^{A_{ij}} \tag{3.9}$$

towards $u(A)$, where we take the convention $0^0 = 1$.

Unfortunately, it proves too computationally taxing to maintain the dynamic program as p becomes large. Compare (3.9) with the one row of the Columns-in-Expectation sampler where the ith row contributes $\prod_{j=1}^{n} w_j^{A_{ij}}$. The problem with (3.9) is that the contribution to $u(A)$ of any row of A depends on the rows above it. To combat this we propose an approximate dynamic programming routine that trades an extra rejection step for efficient initialization and per-proposal sampling.

The change we make is to approximate the weight function u. A parameter $d \in \mathbb{N}$ controls the approximation; we choose new weights

$$w(A) = \left(\prod_{j : c_j \le d} \frac{c_j}{(c_j - \sum_{i=1}^{p} A_{ij})} \right) \left(\prod_{j : c_j > d} c_j^{\sum_{i=1}^{p} A_{ij}} \right).$$

We can find an efficient dynamic program for sampling proportionally to w because the dependence that we discussed in the previous paragraph is limited to only columns with small margins. The efficiency gain is akin to that of Miller and Harrison's dynamic program when $\max_j c_j \le d$. Generating one proposal with our algorithm takes $O(mn + p^2 n^2 \log n \log^2 m)$ time and initialization, which must be done only once, takes $O(pn^{d+1}(m^2 n + mn^2) + pn^{2d+3} \log m)$ time. See Theorems 53 and 54 for details.

We begin by introducing the algorithm with a simple example in the next section. It is followed by a detailed description of the sampling algorithm and proof of its correctness in Section 3.7.2 and the two ensuing sections. Section 3.7.5 describes the initialization procedure.

3.7.1 A motivating example

We first illustrate the idea with an example. Suppose that $n = m = 1001$ and $r = c = (500, 1^{(1000)})$. The Configuration Model rejection sampler will reject nearly every proposal on this instance because of multiple edges between the two large degree vertices. The dynamic programming algorithm of

Miller and Harrison is not practical because the maximum margin is very large.[3] Finally, Bezáková, Sinclair, Stefanovic, and Vigoda [18] have shown that the Chen, Diaconis, and Holmes sequential importance sampling algorithm [33] misestimates $|\Sigma_{r,c}|$ by an exponential factor for margins like these. But, this is an easy instance!

We will be sampling rows of the table proportionally to the weights described in the alternative interpretation of the Configuration Model, Section 3.4.2, conditioned that every entry is 0 or 1, which is just a fancy way of saying that we sample uniformly from $\Sigma_{r,c}$. This is accomplished for the first row with Algorithm 5, the single row sampling algorithm used by the Columns-In-Expectation sampler. The sampling weights are just the margins c. For the remaining rows, we use a sample from the Configuration Model.

One easily computes that there are $\binom{1000}{500}$ ways that the first row of the matrix can be assigned with $A_{11} = 0$, and each has weight $1^{500} = 1$. There are $\binom{1000}{499}$ ways that the first row can be assigned so that $A_{11} = 1$, and each has weight $500 \cdot 1^{499} = 500$. That is enough information to sample the first row. Let X denote the random matrix. We set $X_{11} = 1$ with probability

$$P(X_{11} = 1) = \frac{500\binom{1000}{499}}{500\binom{1000}{499} + \binom{1000}{500}},$$

and then select $500 - X_{11}$ distinct indices $j_1, j_2, \ldots \geq 2$ uniformly at random and set $X_{1j_\ell} = 1$ for each of them.

It remains to assign the other 1000 rows of X. Those rows form a 1000×1001 submatrix with every row margin 1 and (random) column margins $(c_1 - X_{11}, c_2 - X_{12}, \ldots, c_n - X_{1n})$. The column margins are some ordering of either $(500, 1^{(500)}, 0^{(500)})$ or $(499, 1^{(501)}, 0^{(499)})$, depending on the outcome of the first row of X. Whichever it is, we will sample the remaining submatrix with the Configuration Model sampler. Because every row margin is 1, the algorithm will always accept, and the completed table is always a member of $\Sigma_{r,c}$.

[3] However, one can modify their algorithm to to sample in polynomial time when there is a constant value d such that there are at most d vertices of degree larger than d. After this introductory example, we will allow more vertices of large degree.

Let $A \in \Sigma_{r,c}$ and let us compute the probability of the event $X = A$. If $A_{11} = 1$ then from the previous paragraph and Equation 3.1 we find

$$P(X = A) = \frac{500\binom{1000}{499}}{500\binom{1000}{499} + \binom{1000}{500}} \cdot \frac{1}{\binom{1000}{499}} \cdot \frac{499!(1!)^{501}(1!)^{1000}}{1000!},$$

and if $A_{11} = 0$ then

$$P(X = A) = \frac{\binom{1000}{500}}{500\binom{1000}{499} + \binom{1000}{500}} \cdot \frac{1}{\binom{1000}{500}} \cdot \frac{500!(1!)^{500}(1!)^{1000}}{1000!}.$$

Cancellations reveal that these are the same and independent of A, so X is a uniformly random sample.

We can do the same far any set of margins r and c. The first row is sampled with the dynamic programming algorithm, where the weights are c. Next we attempt to complete the matrix with one proposal from the Configuration Model sampler. If the result is binary, then it is a uniform sample from $\Sigma_{r,c}$. If it is not binary, then we begin again by resampling the first row.

The reason this approach works is that there is a seamless transition between the dynamic programming with weights and the Configuration Model. The sampling distributions of the two models are synchronized. It happens because of the way the weights are chosen. For a particular matrix $A \in \Sigma_{r,c}$, the probability that the first row of a random matrix from this hybrid model is the same as the first row of A is proportional to $\prod_j c_j^{A_{1j}}$ and, using Equation 3.1, the probability that the remaining $(m-1) \times n$ submatrix is the same as the lower $(m-1) \times n$ submatrix of A is exactly

$$\frac{(\prod_{i=2}^{m} r_i!)\left(\prod_j (c_j - A_{1j})!\right)}{(\sum_{i=2}^{m} r_i)!}.$$

Hence, the binary matrix A is chosen with probability proportional to

$$\prod_j c_j^{A_{1j}}(c_j - A_{1j})! = \prod_j c_j!,$$

that is uniformly.

The next sections present our full algorithm. It improves over the present example in two ways. The first improvement is that the algorithm samples $p \geq 1$ binary rows with dynamic programming before transitioning to the Configuration Model. The natural generalization is that these rows should be sampled proportionally to $u(A)$ so that the hybrid model samples a matrix $A \in \Sigma_{r,c}$ proportionally to

$$u(A)\left(c - \sum_{i=1}^{p} A_i\right)! = c!,$$

which is uniform.

The running time of the dynamic programming algorithm that samples the first p rows grows exponentially in p. That is a problem. Our second improvement is to introduce an approximation to the table weights so that the dynamic program can be computed efficiently. The approximation controls the initialization and per-proposal time complexity at the cost of an additional rejection step that is required to preserve the uniformity of the sampling distribution.

3.7.2 Sampling overview

This section provides an overview of the Hybrid sampler. The Hybrid sampler works in two stages: sampling first the top p rows of the table with dynamic programming and then the remaining $m - p$ rows with the Configuration Model sampler. The next two sections describe the dynamic programming procedure that accomplishes the first stage sampling, and Section 3.7.5 describes the initialization of that procedure.

Recall $r|_i = (r_{i+1}, \ldots, r_m)$, $r|^i = (r_1, r_2, \ldots, r_i)$, and $A|^i$ the matrix consisting of the top i rows of the matrix A. We will generate a sample for which every entry in the first p rows is binary; of course, the row margins can be rearranged so that any p rows are first. Let $D = \max_j c_j$. Let

$$\mathcal{F}_i = \left\{A|^i : A \in \Sigma_{r,c}\right\}$$

be the set of $i \times n$ binary matrices that are feasible to be completed to a matrix in $\Sigma_{r,c}$ by appending $m - i$ binary rows. $\mathcal{F}_0$ contains a single "empty" matrix and $\mathcal{F}_m = \Sigma_{r,c}$.

Recall the vector $V(c) \in \mathbb{N}^D$ with $V(c)_\ell = \#\{j \in [n] : c_j = \ell\}$, for $\ell = 1, 2, \ldots, D$. For a matrix A we define the vector $L(A) \in \mathbb{N}^D$ where

$$L(A)_\ell = V(c - \sum_i A_i)_\ell = \#\{j : c_j - \sum_i A_{ij} = \ell\}.$$

L is defined so that if $A \in \Sigma_{r,c}$ then $L(A|^i) = V(\sum_{k=i+1}^m A_k)$. If we choose the first i rows of a matrix to be $A \in \mathcal{F}_i$ then $L(A)$ describes the column margins that must be satisfied by the submatrix consisting of the remaining $m - i$ rows. For $i = 0, 1, \ldots, p$, let

$$\mathcal{L}_i = \{L(A) : A \in \mathcal{F}_i\}.$$

In particular, $v \in \mathcal{L}_i$ if and only if $\Sigma_{r|_i,c'} \neq \emptyset$ for $c' = (1^{(v_1)}, 2^{(v_2)}, \ldots, D^{(v_D)})$. In the case $i = 0$ notice that $\mathcal{L}_0 = \{V(c)\}$. We can test the membership of a matrix in $\mathcal{F}_i$ or a vector in $\mathcal{L}_i$ with the Gale-Ryser criteria described in Section 3.2.

For each matrix $A \in \mathcal{F}_i$ we define the weight

$$u(A) = \frac{c!}{(c - \sum_{k=1}^i A_k)!}.$$

We choose to sample proportionally to u in order to synchronize the two sampling stages.

Our algorithm begins by sampling a random binary matrix $X' \in \mathcal{F}_p$. X' is distributed so that for each $A' \in \mathcal{F}_p$

$$P(X' = A') \propto u(A).$$

Let $C_j = c_j - \sum_i X'_{ij}$, and notice that $C_j \geq 0$ because of the definition of u and $\sum_{j=1}^n C_j = \sum_{i=p+1}^m r_i$ because $A \in \mathcal{F}_p$. Thus, the vectors $r|_p$ and C are a valid instance for the Configuration Model

sampler. The next step is to sample a random matrix $X'' \in \Gamma_{r|_p,C}$ using the Configuration Model sampler. X'' is conditionally independent of X' given its column sums. If X'' is binary, then we accept the sample

$$X = \begin{pmatrix} X' \\ X'' \end{pmatrix} \in \Sigma_{r,c}.$$

Theorem 50. *The distribution of the random matrix X is uniform with support $\Sigma_{r,c}$.*

Proof. We get $X \in \Sigma_{r,c}$ as the row margins of $X|^p$ are correct because $X|^p \in \mathcal{F}_p$ and the margins for the columns and remaining rows are correct because $X|_p$ is drawn from the Configuration Model with row and column margins $r|_p$ and $(c - \sum_{i=1}^{p} X_i)$, respectively. Given any matrix $A \in \Sigma_{r,c}$ partitioned as above into A' and A'', let $c''_j = \sum_i A''_{ij} = c_j - \sum_i A'_{ij}$. We have

$$\begin{aligned} P(X = A) &= P(X' = A')P(X'' = A''|X' = A') \\ &= \frac{u(A')}{\sum_{M \in \mathcal{F}_p} u(M)} P(X'' = A''|X' = A') \\ &= \frac{c!}{(c - c')! \sum_{M \in \mathcal{F}_p} u(M)} \cdot \frac{(r|_p)!c''!}{\left(\sum_{i=p+1}^{m} r_i\right)!} \\ &= \frac{c!(r|_p)!}{\left(\sum_{M \in \mathcal{F}_p} u(M)\right)\left(\sum_{i=p+1}^{m} r_i\right)!}, \end{aligned}$$

where we have used the formula 3.1 for $P(X'' = A''|X' = A')$. This probability is independent of A; thus, X is uniformly distributed on $\Sigma_{r,c}$. □

The next section describes an inefficient algorithm for sampling X', though it includes almost all of the technology needed for our final algorithm. Section 3.7.4 completes the description of the hybrid sampling algorithm. It is followed by a description of the algorithm's initialization procedure.

3.7.3 Inefficient but exact

We begin by describing a straightforward but inefficient extension of Miller and Harrison's dynamic programming algorithm to sample X' one row at a time. Recall that the running time for Miller and Harrison's algorithm is exponential in maximum column margin. This algorithm has a similar time complexity, typically exponential in p, so it is not practical when p is large. In the next section we will reduce its time complexity to polynomial by allowing the algorithm to approximate some parts of the computation.

The algorithm iteratively samples one row at a time. It is similar to the Miller and Harrison's algorithm, described in Section 3.6, but there are two main differences. This algorithm only samples $p \le m$ rows of the table and it incorporates weights to the coordinates of the table.

For each vector $v \in \mathcal{L}_i$ and $1 \le i \le p$ we define

$$U_{i,v} = \sum_{\substack{A \in \mathcal{F}_i \\ L(A)=v}} u(A),$$

with the convention that $U_{0,V(c)} = 1$. For convenience we let $U_{i,v} = 0$ if $i \notin \{0, 1, \ldots, p\}$ or $v \notin \mathcal{L}_i$. In particular, $U_{0,v} = 0$ if $v \ne V(c)$ because $\mathcal{L}_0 = \{V(c)\}$.

We will find a recursion for U and use it to calculate the needed marginal probabilities. First, a quick fact. For any two vectors $u, v \in \mathbb{N}^D$ of the same dimension there is exactly one solution x to the equation $u \setminus x = v$. Indeed, $v_D = u_D - x_D$ hence $x_D = u_D - v_D$. If $1 \le k \le D-1$ we have $v_k = u_k - x_k + x_{k+1}$ hence $x_k = x_{k+1} + u_k - v_k$, so the fact follows by induction.

For $x \in \mathbb{R}^D$ let $\delta(x) = \prod_{k=1}^{D} k^{x_k}$.

Lemma 51. *For $i \ge 1$ and $v \in \mathcal{L}_i$ if $U_{i,v} > 0$ then*

$$U_{i,v} = \sum_{\substack{y \in \mathcal{L}_{i-1} \\ x : v = y \setminus x}} U_{i-1,y} \binom{y}{x} \delta(x). \tag{3.10}$$

Proof. Replacing U by its definition, (3.10) is equivalent to

$$\sum_{\substack{A\in\mathcal{F}_i\\ L(A)=v}} u(A) = \sum_{\substack{y\in\mathcal{L}_{i-1}\\ x:v=y\setminus x}} \sum_{\substack{A'\in\mathcal{F}_{i-1}\\ L(A')=y}} u(A')\binom{y}{x}\delta(x).$$

We first establish an equivalence relation on the terms of the left side. The lemma is proved by showing that each term on the right side is the sum of the terms in some equivalence class from the left and every class is represented exactly once. The proof is technical, but the main idea is not. Every matrix $A \in \mathcal{F}_i$ is a matrix $A' \in \mathcal{F}_{i-1}$ and one additional row.

The equivalence relation is defined from a function

$$f : \{A \in \mathcal{F}_i : L(A) = v\} \to \mathcal{F}_{i-1} \times \mathbb{N}^D$$

that maps $A \mapsto (A', x)$ as follows. The matrix $A' = A|^{i-1}$ is the first $i-1$ rows of the i-row matrix A and the vector x has coordinates $x_k = |X_k|$ where

$$X_k = \{j \in [n] : c_j - \sum_{\ell=1}^{i-1} A'_{\ell j} = k \text{ and } A_{ij} = 1\}, \text{ for } k = 1, 2, \ldots, D.$$

The vector x describes the ith, i.e. final, row of A. In particular, x_k counts the number of 1s of the row A_i that lie in columns where $A' = A|^{i-1}$ still needs k additional units to meet its margin.

Let $A \in \mathcal{F}_i$ and $(A', x) = f(A)$, we claim that

1. $0 \le x \le L(A')$ coordinate-wise,
2. $u(A) = u(A')\delta(x)$, and
3. $L(A) = L(A') \setminus x$.

Indeed, $x \geq 0$ is obvious and $x \leq L(A')$ follows because

$$X_k = \{j \in [n] : c_j - \sum_{\ell=1}^{i-1} A'_i = k \text{ and } A_{ij} = 1\} \subseteq \{j \in [n] : c_j - \sum_{\ell=1}^{i-1} A'_i = k\}$$

and the latter set has cardinality $L(A')_k$. For the second part of the claim, notice that

$$\left(c_j - \sum_{k=1}^{i} A_{kj}\right)! \left(c_j - \sum_{k=1}^{i-1} A_{kj}\right)^{A_{ij}} = \left(c_j - \sum_{k=1}^{i-1} A_{kj}\right)!,$$

with the convention that $0^0 = 1$. Hence, it follows that

$$\begin{aligned}
u(A) &= \prod_{j=1}^{n} \frac{c_j!}{(c_j - \sum_{k=1}^{i} A_{kj})!} \\
&= \left(\prod_{j=1}^{n} \frac{c_j!(c_j - \sum_{k=1}^{i-1} A_{kj})^{A_{ij}}}{(c_j - \sum_{k=1}^{i-1} A_{kj})!}\right) \\
&= u(A') \prod_{j=1}^{n} (c_j - \sum_{k=1}^{i-1} A_{kj})^{A_{ij}} \\
&= u(A') \prod_{k=1}^{D} k^{|X_k|} \\
&= u(A')\delta(x).
\end{aligned}$$

Finally, for the third part of the claim. We have for $1 \leq \ell < D$

$$\begin{aligned}
L(A)_\ell &= \#\{j : c_j - \sum_{k=1}^{i} A_{kj} = \ell\} \\
&= \#\{j : c_j - \sum_{k=1}^{i-1} A_{kj} = \ell, A_{ij} = 0\} + \#\{j : c_j - \sum_{k=1}^{i-1} A_{kj} = \ell + 1, A_{ij} = 1\} \\
&= \#\left(\{j : c_j - \sum_{k=1}^{i-1} A_{kj} = \ell\} \setminus X_\ell\right) + |X_{\ell+1}| \\
&= L(A')_\ell - x_\ell + x_{\ell+1} \\
&= (L(A') \setminus x)_\ell.
\end{aligned}$$

The case $\ell = D$ follows similarly, hence $L(A) = L(A') \setminus x$.

The function f induces an equivalence relation on $\mathcal{F}_i$ where $A \sim M$ if and only if $f(A) = f(M)$. If $A \sim M$ then $u(A) = u(M) = u(A')\delta(x)$. The equivalence class containing $A \mapsto (A', x)$ has cardinality $\binom{L(A')}{x}$, hence its total weight is $u(A)\binom{L(A')}{x} = u(A')\binom{L(A')}{x}\delta(x)$. Summing over the equivalence classes gives

$$U_{i,v} \leq \sum_{\substack{y \in \mathcal{L}_{i-1} \\ x : v = y \setminus x}} \sum_{\substack{A' \in \mathcal{F}_{i-1} \\ L(A') = y}} u(A') \binom{y}{x} \delta(x) = \sum_{\substack{y \in \mathcal{L}_{i-1} \\ x : v = y \setminus x}} U_{i-1,y} \binom{y}{x} \delta(x),$$

because all of the terms are nonnegative.

It remains to show that every positive term A', x in the center expression above is the image under f of some $A \in \mathcal{F}_i$, as this implies the reverse inequality above. Indeed, $\binom{y}{x} \neq 0$ implies $0 \leq x \leq y$ coordinate-wise. We can create an ith row for A' by taking the left-most x_k columns among the y_k columns with $c_j - \sum_{\ell=1}^{i-1} A'_{\ell j} = k$ and assigning them 1s, for each k, and assigning 0s elsewhere. Call the resulting matrix A. By definition $L(A) = L(A') \setminus x = v$, hence $A \in \mathcal{F}_i$, and we have defined A so that $f(A) = (A', x)$. □

Now we describe how to use the table U to sample from $\mathcal{F}_p$ proportionally to u. Sampling from $\mathcal{F}_p$ is done in two steps. Let X denote the $p \times n$ matrix under construction. First, we determine the vectors $L(X|^i)$ for $i = p, p-1, p-2, \ldots, 1$. Second, we determine the rows X_i in order $i = 1, 2, \ldots, p$.

$L(X)$ is sampled from the distribution

$$P(L(X) = v) = \frac{U_{p,v}}{\sum_{v' \in \mathcal{F}_p} U_{p,v'}}.$$

Let $v^{(p)} = L(X)$ denote the outcome. Next we sample $L(X|^i)$ for $i = p-1, p-2, \ldots, 1$ with probability

$$P(L(X|^i) = y) = \frac{U_{i,y}}{U_{i,v^{(i+1)}}} \binom{y}{x} \delta(x), \tag{3.11}$$

where x is the (unique) solution to $v^{(i+1)} = y \setminus x$ and $v^{(i)}$ is the outcome of $L(X|^i)$. Let $v^{(0)} = V(c)$ and let $x^{(i)}$ be the solution to $v^{(i)} = v^{(i-1)} \setminus x$, for $i = 1, 2, \ldots, p$. To sample the ith row (starting with $i = 1$) we choose at random one of the $\binom{v^{(i-1)}}{x^{(i)}}$ possible ways to assign X_i that leaves $L(X|^i) = v^{(i)}$.

After we check the distribution of X, we are ready to get at the problematic efficiency of this method for sampling. Let $A \in \mathcal{F}_p$; our aim is to show that $P(X = A) \propto u(A)$. For $i = 1, 2, \ldots, p$, let $a^{(i)} \in \mathbb{N}^D$ be the vector with

$$a_\ell^{(i)} = \#\left\{ j \in [n] : c_j - \sum_{k=1}^{i-1} A_{kj} = \ell \text{ and } A_{ij} = 1 \right\}$$

so that $L(A|^i) = L(A|^{i-1}) \setminus a^{(i)}$, for $i \in [p]$. What we have is that $\{X = A\}$ is the event that, for each row i, $L(X|^i) = L(A|^i)$ and the random assignment of the coordinates of row i coincide with A_i. In other words,

$$P(X = A) = P\left(L(X|^i) = L(A|^i), \text{ for } i \in [p]\right) \cdot P\left(X_i = A_i, i \in [p] \middle| L(X|^i) = L(A|^i), \text{ for } i \in [p]\right). \tag{3.12}$$

For the first term, it follows from (3.11) that for $k > i+1$ the vector $L(X|^i)$ is conditionally independent of $L(X|^k)$ given $L(X|^{i+1})$. Thus, by conditioning we get

$$\begin{aligned}
P\left(L(X|^i) = L(A|^i), \text{ for } i \in [p]\right) =& P\big(L(X|^p) = L(A|^p)\big) \prod_{i=1}^{p-1} P\left(L(X|^i) = L(A|^i) \middle| L(X|^{i+1}) = L(A|^{i+1})\right) \\
=& \frac{U_{p,L(A)}}{\sum_v U_{p,v}} \prod_{i=1}^{p-1} \frac{U_{i,L(A|^i)} \binom{L(A|^i)}{a^{(i+1)}} \delta(a^{(i+1)})}{U_{i+1,L(A|^{i+1})}} \\
=& \frac{U_{1,V(c)}}{\sum_v U_{p,v}} \prod_{i=1}^{p-1} \binom{L(A|^i)}{a^{(i+1)}} \delta(a^{(i+1)}) \\
=& \frac{1}{\sum_v U_{p,v}} \prod_{i=0}^{p-1} \binom{L(A|^i)}{a^{(i+1)}} \delta(a^{(i+1)}),
\end{aligned}$$

where the last equality follows because $L(A|^0) = V(c)$ and $U_{1,v} = \binom{V(c)}{x}\delta(x)$ for $v = V(c) \setminus x$.

For the second term from (3.12), we have

$$P\left(X_i = A_i, i \in [p] \middle| L(X|^i) = L(A|^i), \text{ for } i \in [p]\right) = \prod_{i=1}^{p} P\left(X_i = A_i \middle| X|^{i-1} = A|^{i-1}, L(X|^i) = L(A|^i)\right)$$
$$= \prod_{i=1}^{p} \frac{1}{\binom{L(A|^{i-1})}{a^{(i)}}}$$

Combining the two terms, we get

$$P(X = A) = \frac{\prod_{i=1}^{p} \delta(a^{(i)})}{\sum_v U_{p,v}} = \frac{u(A)}{\sum_v U_{p,v}},$$

from the definition of δ and $a^{(i)}$. That is exactly the distribution that we wanted, X is sampled from $\mathcal{F}_p$ proportionally to the weight u.

Now for the catch, the problem with this approach is that there may be exponentially many vectors in $\mathcal{L}_i$, and thus exponentially many terms on the right-hand side of (3.10). There is no clear way to organize the computation that avoids the exponential sized sum. The algorithm captures the same efficiency gains as Miller and Harrison's algorithm [82] in that the table only depends on the unordered remaining margins $L(X|^i)$. However, initialization time is exponential in p, and although $p \leq \max_j c_j$, we do not want to be restricted to only choosing a very small value for p.

3.7.4 Efficient but approximate

We now describe a rejection sampler for sampling X from $\mathcal{F}_p$ proportionally to u. The first step is to choose a value for d, the parameter we first mentioned at the beginning of Section 3.7. The running time of the algorithm will be exponential in d and increasing d will decrease the rejection probability.

Let $\mathcal{D} = \{j : c_j \leq d\}$ and $\overline{\mathcal{D}} = [n] \setminus \mathcal{D}$. Given a binary matrix A let

$$w(A) = \left(\prod_{j \in \mathcal{D}} \frac{c_j!}{(c_j - \sum_i A_{ij})!}\right)\left(\prod_{j \notin \mathcal{D}} c_j^{\sum_i A_{ij}}\right).$$

We define $\tilde{L}(A) \in \mathbb{N}^d$ to be the vector with coordinates $\tilde{L}(A)_k = \#\{j \in \mathcal{D} : c_j - \sum_i A_{ij} = k\}$. Along the lines of the earlier (inefficient) algorithm, we define $\tilde{\mathcal{L}}_i = \{\tilde{L}(A) : A \in \mathcal{F}_i\}$.

Initially we will sample a $p \times n$ matrix from the larger class of matrices $\tilde{\mathcal{F}}_p$ where

$$\tilde{\mathcal{F}}_i = \{A \in \Sigma_{r|^i} : \tilde{L}(A) \in \tilde{\mathcal{L}}_i\}.$$

$\tilde{\mathcal{F}}_i$ is the set of $i \times n$ binary matrices with row margins $r_1, r_2, \ldots, r_i$ where the (unordered) column margins for columns in $\mathcal{D}$ are the same as for some matrix in $\mathcal{F}_i$. In general $\mathcal{F}_i \subsetneq \tilde{\mathcal{F}}_i$. An easy example to see that the inclusion is strict is to take $i = m$. We have $\mathcal{F}_m = \Sigma_{r,c}$, all column sums are as desired, whereas

$$\tilde{\mathcal{F}}_m = \{A \in \Sigma_r : \sum_i A_{ij} = c_j \text{ for all } j \in \mathcal{D}\},$$

i.e. only columns in $\mathcal{D}$ must have the correct sum.

Before we go into more details, here is an overview of the algorithm. The first step is to sample a matrix X' from $\tilde{\mathcal{F}}_p$ so that $P(X' = A)$ is proportional to $w(A)$. Quite simply, we get

$$u(A) = w(A) \prod_{j \notin \mathcal{D}} \frac{c_j!}{c_j^{\sum_i A_{ij}} (c_j - \sum_i A_{ij})!} = w(A) \prod_{j \notin \mathcal{D}} \left(\frac{c_j}{c_j} \cdot \frac{c_j - 1}{c_j} \cdots \frac{c_j - \sum_i A_{ij}}{c_j} \right).$$

The second step is the extra accept/reject step. We accept X' with probability

$$\frac{u(X')}{w(X')} = \prod_{j \notin \mathcal{D}} \left(\frac{c_j}{c_j} \cdot \frac{c_j - 1}{c_j} \cdots \frac{c_j - \sum_i X'_{ij}}{c_j} \right).$$

Conditionally given acceptance, X' is a sample from $\tilde{\mathcal{F}}_p$ with distribution $P(X' = A) \propto u(A)$. Now, we sample a matrix X'' from the Configuration Model with row margins $r|_p$ and column margins $c - \sum_i X'_i$. If X'' is binary then we accept the sample $X = \binom{X'}{X''} \in \Sigma_{r,c}$. It should be clear that if we condition on $X' \in \mathcal{F}_p$ then X is uniformly distributed on $\Sigma_{r,c}$ as a consequence of Theorem 50. If instead $X' \in \tilde{\mathcal{F}}_p \setminus \mathcal{F}_p$ then there is no matrix in $\Sigma_{r,c}$ with X' as the first p rows. That means that

$P(X'' \text{ is binary}) = 0$. Hence, the distribution of X is uniform on $\Sigma_{r,c}$ without conditioning.

Our next step is to sort out a recursion that we can use to sample proportionally to the weights w. For $i \in \{0, 1, \ldots, p\}$ and $v \in \tilde{\mathcal{L}}_i$ let

$$W_{i,v} = \sum_{\substack{A \in \tilde{\mathcal{F}}_i \\ \tilde{L}(A) = v}} w(A).$$

In particular, $W_{0,v} = 1$ if $v_k = \#\{j \in \mathcal{D} : c_j = k\}$ and $W_{0,v} = 0$ otherwise. Let

$$T_\ell = \sum_{\substack{a \in \{0,1\}^{\overline{\mathcal{D}}} \\ \sum a_k = \ell}} \prod_{j \notin \mathcal{D}} c_j^{a_j}$$

be the total weight, when the weights are c, of all binary vectors in $\mathbb{N}^{\overline{D}}$ with ℓ 1s. A dynamic program for computing T was one of the first items in our discussion of the Columns-in-Expectation sampler of Section 3.5, specifically Equation 3.2.

Notice that $|\tilde{\mathcal{L}}_i| \le n^d$. This solves the efficiency problem for sampling algorithm, because we get to choose d. When we choose d to be a constant we can generate proposals in polynomial time.

Lemma 52. *For $1 \le i \le p$ and $v \in \tilde{\mathcal{L}}_i$ if $W_{i,v} > 0$ then*

$$\begin{aligned} W_{i,v} &= \sum_{\substack{y \in \tilde{\mathcal{L}}_{i-1} \\ x : v = y \setminus x}} W_{i-1,y} \binom{y}{x} y^x \sum_{\substack{a \in \{0,1\}^{\overline{\mathcal{D}}} \\ \sum a_k = r_i - \sum_k x_k}} \prod_{j \notin \mathcal{D}} c_j^{a_j} \\ &= \sum_{\substack{y \in \tilde{\mathcal{L}}_{i-1} \\ x : v = y \setminus x}} W_{i-1,y} \binom{y}{x} \delta(x) T_{r_i - \sum_k x_k}. \end{aligned} \tag{3.13}$$

Proof. As in the proof of Lemma 51 we use a function to define an equivalence relation on

$$\{A \in \tilde{\mathcal{F}}_i : \tilde{L}(A) = v\}.$$

The relation has the property that each equivalence class is represented by a positive term of (3.13) and vice versa.

The function is

$$g : \{A \in \tilde{\mathcal{F}}_i : \tilde{L}(A) = v\} \to \tilde{\mathcal{F}}_{i-1} \times \mathbb{N}^d \times \{0,1\}^{\overline{\mathcal{D}}}$$

that maps a matrix A to (A', x, a) where $A' = A|^{i-1}$ is the first $i-1$ rows (out of i total) of A, x is the vector that solves $\tilde{L}(A) = \tilde{L}(A') \setminus x$, and a is the restriction of A_i to the columns $\overline{\mathcal{D}}$. As with f in the proof of Lemma 51, g induces an equivalence relation on $\tilde{\mathcal{F}}_i$ where two matrices are equivalent if g maps them to the same triple. Within the equivalence class containing A there are $\binom{L(A')}{x}$ matrices and each has equal weight

$$w(A')\delta(x) \prod_{j \notin \mathcal{D}} c_j^{a_j} = w(A).$$

The claim now follows by summing over all equivalence classes so that $\tilde{L}(A) = v$. □

Sampling a matrix $X \in \Sigma_{r,c}$ is done in four steps.

1. Determine the partial margins $\tilde{L}(X|^i)$, for $i = p, p-1, \ldots, 1$.
2. Determine the row X_i, for $i = 1, 2, \ldots, p$.
3. Accept with probability $u(X|^p)/w(X|^p)$ or reject and start over.
4. Sample $X|_p$ with the Configuration Model, if $X|_p$ is not binary then reject and start over.

If the third step is accept, rather than reject, and the final matrix is binary then we accept it as a uniform sample from $\Sigma_{r,c}$. Algorithm 9 describes the entire procedure. The analysis presumes that $O(\log \ell)$ time required to generate a random integer in the range $[\ell]$. Furthermore, the stated time complexity is a count of bit operations, rather than arithmetic operations. This is for two reasons. First, Miller and Harrison's analysis is also based on bit operations so these time bounds

are directly comparable to theirs. Second, the numbers involved can grow very large, so that performing arithmetic operations is expensive. Indeed, we use the easy bound of

$$\sum_v W_{i,v} \leq 2^{pn} m^{pn}$$

to bound the total weight of all of the tables. The time required to add two integers is linear in the number of bits and the time to multiply them is quadratic (with the algorithm taught in grade school). Hence, just multiplying two large values may take on the order of $(pn \log m)^2$ time.

Theorem 53. *Conditionally given that Algorithm 9 does not reject, the matrix that it outputs is uniformly distributed on $\Sigma_{r,c}$. When d is constant, the time to generate one proposal or abort is $O(mn + p^2 n^2 \log n \log^2 m)$, presuming that each entry in W can be accessed in $O(1)$ time.*

Proof. First, we prove that the algorithm is well defined. It has already been observed that the acceptance probability $0 \leq u(A)/w(A) \leq 1$, so we only need to show that the sampling instance given to the Configuration Model sampler, let us call the margins r', c', is well defined. That is the case when $r', c' \geq 0$ and $\sum_i r' = \sum_j c'$. It is fine if $\Sigma_{r',c'} = \emptyset$; in this case, the Configuration Model is well defined but obviously any sample from it will not be binary, so the algorithm will abort at the end. Let $X' \in \tilde{\mathcal{F}}_p$ be the initial proposed matrix. It follows from the definition of u that $P(E_1|X') = u(X')/w(X')$ is positive if and only if $c' = c - \sum_{i=1}^{p} X_i \geq 0$. By construction $\sum_j X_{ij} = r_i$, for $i \leq p$, i.e. $X|^p$ has the correct row margins, so

$$\sum_j c'_j = \sum_j \left(c_j - \sum_{i=1}^{p} X_{ij} \right) = \sum_j c_j - \sum_{i=1}^{p} r_i = \sum_{i=p+1}^{m} r_i = \sum_i r'_i.$$

Thus, $r', c' \geq 0$ and $\sum_i r'_i = \sum_j c'_j$, so the algorithm is well defined.

In order to prove that the sampling distribution is correct it is enough to prove that any output matrix is in $\Sigma_{r,c}$ and every matrix in $\Sigma_{r,c}$ is output with equal probability. Let E be the event that the algorithm outputs a matrix. Define a matrix-valued random variable X to be equal

to the output when E occurs, and 0 otherwise. Let E_1 be the event that algorithm accepts on the first rejection step.

We claim that E implies $X \in \Sigma_{r,c}$. Indeed, from the definition of the Configuration Model and because the algorithm is well defined we get that X has row margins r and column margins c. Furthermore, E implies $X|_p$ is binary and $X|^p$ is always binary. Thus, $\{X \in \Sigma_{r,c}\} \subseteq E$.

Next, we claim that every matrix in $\Sigma_{r,c}$ is output with the same probability. Some basic probability reveals

$$\begin{aligned} P(X = A|E_1) &= P\left(X|^p = A|^p \middle| E_1\right) \cdot P\left(X|_p = A|_p \middle| X|^p = A|^p, E_1\right) \\ &= \frac{P(X|^p = A|^p)}{P(E_1)} \cdot P\left(E_1 \middle| X|^p = A|^p\right) \cdot P\left(X|_p = A|_p \middle| X|^p = A|^p, E_1\right). \end{aligned}$$

The numerator of the first term is the sampling distribution defined by the table W, the second term is the probability of acceptance, and the third term is the probability that the Configuration Model produces $A|_p$. Substituting in for these values and using the definition of u we get

$$\begin{aligned} P(X = A|E_1) &= \frac{w(A|^p)}{P(E_1)\sum_{v\in\tilde{\mathcal{F}}_p} W_{p,v}} \cdot \frac{u(A|^p)}{w(A|^p)} \cdot \frac{(r|_p)!(c - \sum_{i=1}^p A_i)!}{\left(\sum_{i=p+1}^n r_i\right)!} \\ &= \frac{\frac{c!}{(c-\sum_{i=1}^p A_i)!}(r|_p)!(c - \sum_{i=1}^p A_i)!}{P(E_1)\left(\sum_{v\in\tilde{\mathcal{F}}_p} W_{p,v}\right)\left(\sum_{i=p+1}^n r_i\right)!} \\ &= \frac{c!(r|_p)!}{P(E_1)\left(\sum_{v\in\tilde{\mathcal{F}}_p} W_{p,v}\right)\left(\sum_{i=p+1}^n r_i\right)!}, \end{aligned}$$

which does not depend on A. This proves that the sampling is uniform.

Finally, we bound the running time. We begin with the time required to perform individual arithmetic operations and perform the needed random sampling. Trivially, $w(A) \le m^{in}$, for all $A \in \tilde{\mathcal{F}}_i$, hence $W_{i,v} \le \sum_v W_{i,v} \le 2^{in}m^{in}$. Hence, the dynamic programming implementation requires $O(pn \log m)$ operations to perform any addition related to the table and $O(p^2n^2 \log^2 m)$ operations for any multiplication, using the grade school algorithms. The values of $\binom{y_k}{x_k}$, k^{x_k}, and

T in each term of (3.13) can be precomputed. Thus, computing a single term of (3.13) from its components requires only $2d + 1 = O(1)$ multiplications.

Random sampling is done with the following algorithm. Suppose we are given nonnegative integers $a_1, a_2, \ldots, a_N$ and $a = \sum a_i$, and we wish to sample $i \in [N]$ with probability a_i/a. We first draw a random integer $Z \in [a]$ and then check $\sum_{i=1}^{j} a_i < Z$ for each $j = 1, 2, \ldots, N$ until the first time, call it j^*, that the inequality fails. Then the sample is j^*. The sampling can be with only $O(\log N)$ integer comparisons as follows. We precompute, i.e. during initialization, a binary tree where the leaves are $a_1, a_2, \ldots, a_N$ and each node stores the sum of all of the leaves descended from it. This changes the initialization time and space required by the algorithm only by a constant factor. With the tree it is simple to sample with $O(\log N)$ integer comparisons, thus the Each integer comparison requires $O(\log a)$ bit comparisons thus this sampling requires requires $O(N \log a)$ time.

Lets bound the time to sample the ith row. We have $|\tilde{\mathcal{L}}_i| \leq n^{d+1}$ since there are at most n vertices of any degree $0, 1, 2, \ldots, d$. Because each term of (3.13) can be computed in $O(p^2 n^2 \log^2 m)$ time, it requires

$$O(\log n^{d+1}(p^2 n^2 \log^2 m)) = O(p^2 n^2 \log n \log^2 m)$$

time to sample $L(X|^p)$ and it takes no longer for $L(X|^i)$, when $i \leq p$. Completing each of the first p rows requires $O(n)$ time and performing the rejection step requires $O(pn \log m)$ time, both are negligible. Finally, completing the matrix with the Configuration Model requires $O(mn)$ time to sample a random permutation and the same to fill in the matrix. Thus, the total sampling time is $O(mn + p^2 n^2 \log n \log^2 m)$. □

3.7.5 Initialization

As with the initialization of Miller and Harrison's dynamic programming algorithm in Section 3.6.2, the table W is computed using the recursion in Lemma 52. One aspect of that computation is to (efficiently) generate the feasible margins, i.e. the set $\tilde{\mathcal{L}}_i$. Since $|\tilde{\mathcal{L}}_i| \leq n^{d+1}$ it is enough to be able

Algorithm 9 The hybrid algorithm for uniform sampling.

procedure HYBRIDSAMPLE(r, c, p, d)
  Sample $L(X|^p)$ according to $P(L(X|^p) = v) = W_{p,v}/\sum_v W_{p,v}$.
  for $i = p-1, p-2, \ldots, 1$ **do**
    Sample $L(X|^i)$ as $P(L(X|^i = v) = W_{i,v}\binom{v}{x}\delta(x)T_{r_i - \sum_k x_k}/W_{i+1,L(X|^{i+1})}$, where $v \setminus x = L(X|^{i+1})$
  end for
  for $i = 1, 2, \ldots, p$ **do**
    Let x solve $L(X|^{i-1}) \setminus x = L(X|^i)$.
    Sample the values X_{ij}, for $j \in \mathcal{D}$, among the $\binom{L(X|^{i-1}}{x}$ possibilities
    Sample the values X_{ij}, for $j \notin \mathcal{D}$, using T so that $\sum_j X_{ij} = r_i$.
  end for
  Reject with probability $u(X|^p)/w(X|^p)$
  if Reject **then**
    Abort
  end if
  Sample $X|_p$ from the Configuration Model to guarantee row margins r and column margins c
  if X is binary **then**
    return X
  else
    Abort
  end if
end procedure

to test for a given vector v whether $v \in \tilde{\mathcal{L}}_i$ or not.

The main challenge with testing $v \in \tilde{\mathcal{L}}_i$ is that v does not tell us the correspondence to the overall margins. For example, suppose there exists $A \in \Sigma_{r,c}$ with $L(A|^p) = v$. Then we know that $A|_p$ has exactly v_1 column margins, among columns in $\mathcal{D}$, that are equal to 1, but we do not know which columns they are. Indeed, there can be many workable correspondences but we cannot test every possibility in order to find one.

We will first suppose that the correspondence is known and proceed to demonstrate an algorithm for checking $v \in \tilde{\mathcal{L}}_i$ or not. That is, we devise a test to determine for $c' \in \mathbb{N}^{\mathcal{D}}$ whether there exists $A \in \tilde{\mathcal{F}}_i$ such that $c_j - \sum_{k=1}^{i} A_{ij} = c'_j$ for $j \in \mathcal{D}$. After that we will show that it is sufficient to test the membership with a canonical correspondence.

Let c'_j, for $j \in \mathcal{D}$, be the partial margins. We will test whether there exists $A \in \Sigma_{r,c}$ such that $\sum_{i=p+1}^{m} A_{ij} = c'_j$, for all $j \in \mathcal{D}$, using a max-flow computation. Create a graph with $m + n + |\mathcal{D}| + 2$ vertices x_i, for $i \in [m]$; y_j, for $j \in ([n] \setminus \mathcal{D})$; z_{kj}, for $k = 1, 2$ and $j \in \mathcal{D}$; and source

s and sink t. Add directed edges with capacities as shown in Table 3.2.

Edge		Capacity
(s, x_i),	for $i \in [m]$	r_i
(x_i, z_{1j}),	for $i \leq p$, $j \in \mathcal{D}$	1
(x_i, z_{2j}),	for $i > p$, $j \in \mathcal{D}$	1
(x_i, y_j),	for $i \in [m]$, $j \in ([n] \setminus \mathcal{D})$	1
(z_{1j}, t),	for $j \in \mathcal{D}$	$c_j - c'_j$
(z_{2j}, t),	for $j \in \mathcal{D}$	c'_j
(y_j, t),	for $j \in ([n] \setminus \mathcal{D})$	c_j

Table 3.2: Edges and capacities to determine whether there exists a table in $\Sigma_{r,c}$ with intermediate margins c'_j, for $j \in \mathcal{D}$.

An integral maximum s-t flow in this network can be found in $O(|V||E|) = O(m^2n + mn^2)$ time with the algorithm of King, Rao, and Tarjan [72]. If any s-t flow with value $\sum_i r_i = \sum_j c_j$ exists then let A_{ij} be the flow value on edge (x_i, z_{1j}), (x_i, z_{2j}), or (x_i, y_j), whichever edge exists in the graph (exactly one of them exists in the graph). We find that $A \in \Sigma_{r,c}$ and $\sum_{i=p+1}^{m} A_{ij}$ is the total flow into vertex z_{2j} which must be c'_j. The reverse argument shows that if any such matrix exists then the value of a maximum flow is $\sum_i r_i = \sum_j c_j$. Thus, if we know the correspondence with the column margins then we can test in $O(m^2n + mn^2)$ time whether there is any satisfying matrix.

Now for the canonical correspondence. Without loss of generality, let $\mathcal{D} = \{1, 2, \ldots, k\}$ and $c_1 \geq c_2 \geq \cdots \geq c_k$. Let $c'_1 = c'_2 = \ldots = c'_{v_d} = d$, then $c'_{v_d+1} = \cdots = c'_{v_d+v_{d-1}} = d - 1$, until c'_k. The correspondence is c_i with c'_i. One can think of matching each column in $\mathcal{D}$ in decreasing order of their margins with the intermediate margins in decreasing order of their margins. We claim that if there exists a matrix $A \in \Sigma_{r,c}$ with $L(A|^p) = v$ then there is a matrix with the correspondence given.

Indeed, among all matrices $A \in \Sigma_{r,c}$ with $L(A|^p) = v$ choose one with corresponding column margins $c''_j = \sum_{i=p+1}^{m} A_{ij}$, for $j \in [k] = \mathcal{D}$, that has the minimum number of pairs $j < k$ such that $c_j > c_k$ and $c''_j < c''_k$. Let A denote this matrix. If the number of such pairs is 0 then $c'' = c'$ and we are done as A has the correspondence c'. Instead, suppose that there is at least one

such pair $j < k$. We will demonstrate a matrix with fewer such pairs than A. That is a contradiction, hence $c'' = c'$.

Let x and y be the jth and kth columns of A. The setting is $\sum_{i=p+1}^{m} x_i = c''_j$, $\sum_{i=p+1}^{m} y_i = c''_k$, $c''_j < c''_k$, and $c_j > c_k$. Thus there exist at least $c''_k - c''_j$ rows $i > p$ such that $x_i = 0$ and $y_i = 1$. There also exist

$$(c_j - c''_j) - (c_k - c''_k) \geq (c_k - c''_j) - (c_k - c''_k) = c''_k - c''_j$$

rows $i \leq p$ such that $x_i = 1$ and $y_i = 0$. Thus we can form a new matrix by moving $c''_k - c''_j$ 1s from x to y among rows $1, 2, \ldots, p$ and $c''_k - c''_j$ 1s from y to x among rows $p+1, \ldots, m$. The resulting matrix is obviously binary and the row and column margins are unchanged. However, it has at least one fewer inverted pair.

Theorem 54. *For constant d, the table W can be computed in $O(pn^{d+1}(m^2n+mn^2)+p^2n^{2d+4}\log^2 m)$ time, assuming $O(1)$ time access to its entries.*

Proof. We determine the set $\tilde{\mathcal{L}}_i$, for $i = 1, \ldots, p$, as a preprocessing step. It takes $O(m^2n + mn^2)$ operations to test whether $v \in \tilde{\mathcal{L}}_i$. There are $O(pn^{d+1})$ pairs to be tested, hence the total time to determine the sets $\tilde{\mathcal{L}}_i$ is $O(pn^{d+1}(m^2n + mn^2))$.

The table T can be computed in $O(mn)$ time as a preprocessing step. Computing each term on the right hand side of the recursion (3.13) requires $O(1)$ multiplications, so the total time $O(p^2n^2\log^2 m)$. The sum has $O(n^{d+1})$ terms, so it takes $O(p^2n^{d+3}\log^2 m)$ operations to compute $W_{i,v}$ once $W_{i-1,v'}$ is known for all $v' \in \tilde{\mathcal{L}}_{i-1}$. The entire table includes $O(pn^{d+1})$ values, hence the total time to compute W once $\tilde{\mathcal{L}}_i$ is known for $i \in [p]$ is $O(p^3n^{2d+4}\log^2 m)$. Thus the total time to compute the table is $O(pn^{d+1}(m^2n + mn^2) + p^2n^{2d+4}\log^2 m)$. □

3.7.6 Review

There have been lots of technical details, so lets review the basic ideas. We started by separating the matrices into two parts, the top p rows and the bottom $m - p$ rows. By sampling the top p

rows according to a carefully chosen probability distribution we are able to complete the matrix by sampling the bottom $m - p$ rows from the Configuration Model. Conditionally given that the outcome is a binary matrix we are left with a sample from $\Sigma_{r,c}$. If the matrix is not binary, then we must start over with a fresh attempt at sampling.

To make the first stage of the sampling run efficiently we created an approximate dynamic programming algorithm. It allowed us to decrease the time complexity of the sampling and initialization procedures (by choosing d to be smaller) at the cost of increasing the rejection probability. The extra rejection step was necessary to correct the sampling distribution for the approximation. In comparison to Miller and Harrison's dynamic programming, the run time is fixed by the user's choice of d, rather than exponential in the maximum column sum. Versus the configuration model, we expect a lower rejection probability over all. However, this is not entirely clear because of the impact of the added rejection step. Against the polynomial time approximately uniform simulated annealing sampler of Bezáková, Bhatnagar, and Vigoda [17] the asymptotic order of our running time is only faster for $d \leq 3$. On the other hand, samples from this hybrid algorithm are perfectly uniform rather than approximately so.

Chapter 4

Combinatorial interdiction

4.1 Introduction

One way to understand the robustness of a system is to evaluate attack strategies. This naturally leads to *interdiction* problems; broadly, one is given a set of feasible solutions, along with some rules and a budget for modifying the set, with the goal of inhibiting the solution to an underlying nominal optimization problem. A prominent example that nicely highlights the nature of interdiction problems is maximum flow interdiction. Here, the nominal problem is a maximum s-t flow problem. Given is a directed graph $G = (V, A)$ with arc capacities $u : A \to \mathbb{Z}_{>0}$, a source $s \in V$ and sink $t \in V \setminus \{s\}$. Furthermore, each arc has an *interdiction cost* $c : A \to \mathbb{Z}_{>0}$, and there is a global *interdiction budget* $B \in \mathbb{Z}_{>0}$. The goal is to find a subset of arcs $R \subseteq A$ whose cost does not exceed the interdiction budget, i.e., $c(R) := \sum_{a \in R} c(a) \leq B$, such that the value of a maximum s-t flow in the graph $(V, A \setminus R)$ obtained from G by removing R is as small as possible. In particular, if the value of a maximum s-t flow in $G = (V, E)$ is denoted by $\nu((V, E))$, then we can formalize the

problem as follows

$$\min_{R \subseteq A : c(R) \leq B} \nu((V, E \setminus R)).$$

A set $R \subseteq A$ with $c(R) \leq B$ is often called an *interdiction or removal set*. Similarly, one can define interdiction problems for almost any underlying nominal optimization problem.

Interdiction is of practical interest for evaluating robustness and developing attack strategies. Indeed, even the discovery of the famous Max-Flow/Min-Cut Theorem was motivated by a Cold War plan to interdict the Soviet rail network in Eastern Europe [92]. Interdiction has also been studied to find cost-effective strategies to prevent the spread of infection in a hospital [6], to determine how to inhibit the distribution of illegal drugs [96], to prevent nuclear arms smuggling [86], and for infrastructure protection [91, 35], just to name a few applications.

A significant effort has been dedicated to understanding interdiction problems. The list of optimization problems for which interdiction variants have been studied includes maximum flow [95, 96, 88, 100], minimum spanning tree [43], shortest path [9, 71], connectivity of a graph [101], matching [99, 87], matroid rank [64, 66], stable set [13], several variants of facility location [35, 14], and more.

Although one can generate new interdiction problems mechanically from existing optimization problems, there are few general techniques for their solution. The lack of strong exact algorithms for interdiction problems in not surprising in light of the fact that almost all known interdiction problems are NP-hard. However, it is intriguing how little is known about the approximability of interdiction problems. In the context of interdiction problems, the design of approximation algorithms is of particular interest since it often allows accurate estimation of at least the order of magnitude of a potential worst-case impact, which turns out to be a nontrivial task in this context. Polynomial-time approximation schemes (PTASs) are primarily known only when assuming particular graph structures or other special cases. In particular, for planar graphs PTASs have been found for network flow interdiction [88, 100] and matching interdiction [87]. Furthermore, PTASs based

on pseudopolynomial algorithms have been obtained for some interdiction problems on graphs with bounded treewidth [99, 13]. Connectivity interdiction is a rare exception where a PTAS is known without any further restrictions on the graph structure [101]. Furthermore, $O(1)$-approximations are known for interdicting matchings and more generally for some packing interdiction problems [99, 41]. However, for almost all classical polynomial-time solvable combinatorial optimization problems, like minimum spanning tree, shortest path, maximum flows and maximum matchings, there is a considerable gap between the approximation quality of the best known interdiction algorithm and the currently strongest hardness result. In particular, among the above-mentioned problems, only the interdiction of shortest s-t paths is known to be APX-hard, and matching interdiction is the only one among these problems for which an $O(1)$-approximation is known. The best known algorithm for minimum spanning tree interdiction is a $O(\log n)$-approximation [43], where n is the number of vertices in the graph. For network flow interdiction, no approximation results are known, even though only strong NP-hardness is known from a complexity point of view.

Burch et al. [26] decided to go for a different approach to attack the network flow interdiction problem, leading to the currently best known solution guarantee obtainable in polynomial time. Their algorithm solves a linear programming (LP) relaxation to find a fractional interdiction set that lies on an edge of an integral polytope. It is guaranteed that, for any $\alpha > 0$, one of the vertices on that edge is either a budget feasible $(1 + \alpha)$-approximate solution or a super-optimal solution that overruns the budget by at most a factor of $1 + 1/\alpha$. However, one cannot predetermine which objective is approximated and the choice of α biases the outcome. For simplicity we call such an algorithm a 2-pseudoapproximation since, in particular, by choosing $\alpha = 1$ one either gets a 2-approximation or a super-optimal solution using at most twice the budget. In this context, it is also common to use the notion of a (σ, τ)-approximate solution, for $\sigma, \tau \geq 1$. This is a solution that violates the budged constraint by a factor of at most τ, and has a value that is at most a factor of σ larger than the value of an optimal solution, which is not allowed to violate the budget. Hence, a 2-pseudoapproximation is an algorithm that, for any $\alpha > 0$, either returns a $(1 + \alpha, 1)$-approximate

solution or a $(1, 1 + 1/\alpha)$-approximate solution.

The main result of this chapter is a general technique to get 2-pseudoapproximations for a wide set of interdiction problems. To apply our technique we need three conditions on the nomial problem we want to interdict. First, we need to have an LP description of the nomial problem that has a well-structured dual. In particular box-total dual integrality (box-TDI) is sufficient. The precise conditions are described in Section 4.2. Second, the LP description of the nomial problem is a maximization problem whose objective vector only has $\{0, 1\}$-coefficients. Third, the LP description of the nomial problem fulfills a down-closedness property, which we call *w-down-closedness*. This third condition is fulfilled by all independence systems, i.e., problems where a subset of a feasible solution is also feasible, like forests, and further problems like maximum s-t flows. Again, a precise description is given in Section 4.2. In particular, our framework leads to 2-pseudoapproximations for the interdiction of any problem that asks to find a maximum cardinality set in an independence system for which a box-TDI description exists. This includes maximum cardinality independent set in a matroid, maximum cardinality common independent set in two matroids, b-stable sets in bipartite graphs, and more. Furthermore, our conditions also include the maximum s-t flow problem, thus implying the result of Burch et al. [26], even though s-t flows do not form an independence system. Apart from its generality, our approach has further advantages. When interdicting independent sets of a matroid, we can even handle general nonnegative objective functions, instead of only $\{0, 1\}$-objectives. This is obtained by a reformulation of the weighted problem to a $\{0, 1\}$-objective problem over a polymatroid. Also, we can get a 2-pseudoapproximation for interdicting maximum weight independent sets in a matroid with *submodular* interdiction costs. Submodular interdiction costs allow for modeling economies of scale when interdicting. More precisely, the cost of interdicting an additional element is the smaller the more elements will be interdicted. Additionally, our approach can sometimes be refined by exploiting additional structural properties of the underlying optimization problem to obtain stronger results. We demonstrate this by presenting a PTAS for interdicting b-stable sets in bipartite graphs, which is an NP-hard problem. We complete

the discussion of b-stable set interdiction in bipartite graphs by showing that interdicting classical stable sets in bipartite graphs, which are 1-stable sets, can be done efficiently by a reduction to matroid intersection. This generalizes a result by Bazgan, Toubaline and Tuza [13] who showed that interdiction of stable sets in a bipartite graph is easy if all interdiction costs are one.

Organization of the chapter

In Section 4.2, we formally describe the class of interdiction problems we consider, together with the technical assumptions required by our approach, to obtain a 2-pseudoapproximation. Furthermore, Section 4.2 also contains a formal description of our results. Our general approach to obtain 2-pseudoapproximations for a large set of interdiction problems is described in Section 4.3. In Section 4.4 we show how, in the context of interdicting independent sets in a matroid, our approach allows for getting a 2-approximation for general nonnegative weights and submodular interdiction costs. Section 4.5 shows how our approach can be refined for the interdiction of b-stable set interdiction in bipartite graphs to obtain a PTAS. Furthermore, we also present an efficient algorithm for stable set interdiction in bipartite graphs in Section 4.5.

4.2 Problem setting and results

We assume that feasible solutions to the nominal problem, like matchings or s-t flows, can be described as follows. There is a finite set N, and the feasible solutions can be described by a bounded and nonempty set $\mathcal{X} \subseteq \mathbb{R}^N_{\geq 0}$ such that $\text{conv}(\mathcal{X})$ is an integral polytope[1]. For example, for matchings we can choose N to be the edges of the given graph $G = (V, E)$, and $\mathcal{X} \subseteq \{0,1\}^E$ are all characteristic vectors of matchings $M \subseteq E$ in G. Similarly, consider the maximum s-t flow problem on a directed graph $G = (V, A)$, with edge capacities $u : A \to \mathbb{Z}_{>0}$. Here, we can choose $N = A$ and $\mathcal{X} \subseteq \mathbb{R}^N_{\geq 0}$ contains all vectors $f \in \mathbb{R}^N_{\geq 0}$ that correspond to s-t flows.

[1]The discussion that follows also works for feasible sets $\mathcal{X}$ such that $\text{conv}(\mathcal{X})$ is not integral. However, integrality of $\mathcal{X}$ simplifies parts of our discussion and is used to show that our 2-pseudoapproximation is efficient. Furthermore, all problems we consider naturally have the property that $\text{conv}(\mathcal{X})$ is integral.

Furthermore, the nominal problem should be possible to solve by maximizing a linear function w over $\mathcal{X}$. For the case of maximum cardinality matchings one can maximize the linear function with all coefficients being equal to 1. Finally, we assume that we interdict elements of the ground set N, and the interdiction problem can be described by the following min-max mathematical optimization problem:

$$\begin{aligned} \min_{\substack{R\subseteq N:\\ c(R)\leq B}} \max \quad & w^T x \\ & x \in \mathcal{X} \\ & x(e) = 0 \quad \forall e \in R, \end{aligned} \tag{4.1}$$

where $c : N \to \mathbb{Z}_{>0}$ are interdiction costs on N, and $B \in \mathbb{Z}_{>0}$ is the interdiction budget. It is instructive to consider matching interdiction where one can choose N to be all edges and $\mathcal{X} \subseteq \{0,1\}^N$ the characteristic vectors of matchings. Imposing $x(e) = 0$ then enforces that one has to choose a matching that does not contain the edge e which, as desired, corresponds to interdicting e.

Notice that the above way of describing interdiction problems is very general. In particular, it contains a large set of classical combinatorial interdiction problems, like interdicting maximum s-t flows, maximum matchings, maximum cardinality stable sets of a graph, maximum weight forest, and more generally, maximum weight independent set in a matroid or the intersection of two matroids.

Our framework for designing 2-pseudoapproximations for interdiction problems of type (4.1) requires the following three properties, on which we will expand in the following:

1. The objective vector w is a $\{0,1\}$-vector, i.e., $w \in \{0,1\}^N$,
2. the feasible set $\mathcal{X}$ is *w-down-closed*, which is a weaker form of down-closedness that we introduce below, and
3. there is a linear description of the convex hull $\text{conv}(\mathcal{X})$ of $\mathcal{X}$ which is *box-w-DI solvable*. This is a weaker form of being box-TDI equipped with an oracle that returns an integral dual solution to box-constrained linear programs over the description of $\text{conv}(\mathcal{X})$.

In the following we formally define the second and third condition, by giving precise def-

initions of w-down-closedness and box-w-DI solvability. In particular, condition (3), i.e., box-w-DI solvability, describes how we can access the nominal problem.

4.2.1 w-DOWN-CLOSEDNESS

The notion of w-down-closedness is a weaker form of down-closedness. We recall that a set $\mathcal{X} \subseteq \mathbb{R}^N_{\geq 0}$ is down-closed if for any $x \in \mathcal{X}$ and $y \in \mathbb{R}^N_{\geq 0}$ with $y \leq x$ (componentwise), we have $y \in \mathcal{X}$. Contrary to the usual notion of down-closedness, w-down-closedness depends on the $\{0,1\}$-objective vector w.

Definition 55 (w-down-closedness)**.** Let $w \in \{0,1\}^N$. $\mathcal{X} \subseteq \mathbb{R}^N_{\geq 0}$ is *w-down-closed* if for every $x \in \mathcal{X}$ and $e \in N$ with $x(e) > 0$, there exists $x' \leq x$ such that the following conditions hold:

1. $x' \in \mathcal{X}$;
2. $x'(e) = 0$;
3. $w^T x' \geq w^T x - x(e)$.

Notice that if $\mathcal{X} \subseteq \mathbb{R}^N_{\geq 0}$ is down-closed, then it is w-down-closed for any $w \in \{0,1\}^N$, since one can define $x' \in \mathcal{X}$ in the above definition by $x'(f) = x(f)$ for $f \in N \setminus \{e\}$ and $x'(e) = 0$. Similarly, w-down-closedness also includes all independence systems. We recall that an *independence system* over a ground set N is a family $\mathcal{F} \subseteq 2^N$ of subsets of N such that for any $I \in \mathcal{F}$ and $J \subseteq I$, we have $J \in \mathcal{F}$. In other words, it is closed under taking subsets. Typical examples of independence systems include matchings, forests and stable sets. Naturally, an independence system $\mathcal{F} \subseteq 2^N$ can be represented in $\mathbb{R}^N_{\geq 0}$ by its characteristic vectors, i.e., $\mathcal{X} = \{\chi^I \mid I \in \mathcal{F}\}$, where $\chi^I \in \{0,1\}^N$ denotes the characteristic vector of I. Clearly, for the same reasons as for down-closed sets, the set $\mathcal{X}$ of characteristic vectors of any independence system is w-down-closed for any $w \in \{0,1\}^N$.

Hence, many natural combinatorial optimization problems are w-down-closed for any $w \in \{0,1\}^N$, including matchings, stable sets, independent sets in a matroid or the intersection of two matroids. Furthermore, w-down-closedness also captures the maximum s-t flow problem.

Example 56 (w-down-closedness of s-t flow polytope)**.** Let $G = (V, A)$ be a directed graph with

two distinct vertices $s, t \in V$ and arc capacities $u : A \to \mathbb{Z}_{>0}$. Furthermore, we assume that there are no arcs entering the source s, since such arcs can be deleted when seeking maximum s-t flows. The s-t flow polytope $\mathcal{X} \subseteq \mathbb{R}^A_{\geq 0}$ can then be described as follows (see, e.g., [73]):

$$\mathcal{X} = \left\{ x \in \mathbb{R}^A_{\geq 0} \mid x(\delta^+(v)) - x(\delta^-(v)) = 0 \; \forall v \in V \setminus \{s, t\} \right\},$$

where $\delta^+(v), \delta^-(v)$ denote the set of arcs going out of v and entering v, respectively; furthermore, $x(U) := \sum_{a \in U} x(a)$ for $U \subseteq A$. A maximum s-t flow can be found by maximizing the linear function $w^T x$ over $\mathcal{X}$, where $w = \chi^{\delta^+(s)}$, i.e., $w \in \{0,1\}^A$ has a 1-entry for each arc $a \in \delta^+(s)$, and 0-entries for all other arcs. This maximizes the total outflow of s. Notice that the value of a flow $x \in \mathcal{X}$ is equal to $x(\delta^+(s)) - x(\delta^-(s)) = x(\delta^+(s))$, since there are no arcs entering s; this is indeed the total outflow of s.

To see that $\mathcal{X}$ is w-down-closed, let $x \in \mathcal{X}$ and $e \in A$, and we construct $x' \in \mathcal{X}$ satisfying the conditions of Definition 55 as follows. We compute a path-decomposition of x with few terms. This is a family of s-t paths $P_1, \ldots, P_k \subseteq A$ with $k \leq |A|$ together with positive coefficients $\lambda_1, \ldots, \lambda_k > 0$ such that $x = \sum_{i=1}^k \lambda_i \chi^{P_i}$ (see [2] for more details). Let $I = \{i \in [k] \mid e \in P_i\}$, where $[k] := \{1, \ldots, k\}$, and we set $x' = \sum_{i \in [k] \setminus I} \lambda_i \chi^{P_i}$. The flow $x' \in \mathcal{X}$ indeed satisfies the conditions of Definition 55. This follows from the fact that $x(e) = \sum_{i \in I} \lambda_i$, and each path P_i contains precisely one arc of $\delta^+(s)$, hence, $x'(\delta^+(s)) = x(\delta^+(s)) - \sum_{i \in I} \lambda_i$.

Furthermore, notice that a non-empty w-down-closed system $\mathcal{X}$ always contains the zero vector, independent of $w \in \{0,1\}^N$. By w-down-closedness we can go through all elements $e \in N$ one-by-one, and replace x by a vector $x' \in \mathcal{X}$ with $x'(e) = 0$, thus proving that the zero vector is in $\mathcal{X}$.

4.2.2 box-w-DI solvability

To obtain 2-pseudoapproximations for interdiction problems of type (4.1), we additionally need to have a good description of the convex hull $\text{conv}(\mathcal{X})$ of $\mathcal{X}$. The type of description we need is a weaker form of box-TDI-ness together with an efficient optimization oracle for the dual that returns integral solutions, which we call box-w-DI solvability, where "DI" stands for 'dual integral".

Definition 57 (box-w-DI solvability)**.** A description $\{x \in \mathbb{R}^N \mid Ax \leq b, x \geq 0\}$ of a nonempty polytope P is *box-w-DI solvable* for some vector $w \in \{0,1\}^N$ if the following conditions hold:

1. For any vector $u \in \mathbb{R}^N_{\geq 0}$, the following linear program has an integral dual solution if it is feasible:

$$\begin{aligned} \max \quad & w^T x \\ Ax &\leq b \\ x &\leq u \\ x &\geq 0 \end{aligned} \tag{4.2}$$

   Notice that the dual of the above LP is the following LP:

$$\begin{aligned} \min \quad & b^T y + u^T r \\ A^T y + \quad & r \geq w \\ y \quad & \geq 0 \\ r &\geq 0 \end{aligned} \tag{4.3}$$

2. For any $u \in \mathbb{R}^N_{\geq 0}$, one can decide in polynomial time whether (4.2) is feasible. Furthermore, if (4.2) is feasible, one can efficiently compute its objective value and an integral vector $r \in \mathbb{Z}^N_{\geq 0}$ that corresponds to an optimal integral solution to (4.3), i.e., there exists an integral vector y such that y, r is an integral optimal solution to (4.3).

We emphasize that box-w-DI solvability does not assume that the full system $Ax \leq b, x \geq 0$ is given as input. In particular, this is useful when dealing with combinatorial problems whose

feasible set $\mathcal{X} \subseteq \mathbb{R}^N_{\geq 0}$ is such that the polytope $\text{conv}(\mathcal{X})$ has an exponential number of facets, and a description of $\text{conv}(\mathcal{X})$ therefore needs an exponential number of constraints[2]. Since the only access to $\mathcal{X}$ that we need is an oracle returning an optimal integral dual solution to (4.3), we can typically deal with such cases if we have an implicit description of the system $Ax \leq b, x \geq 0$ over which we can separate with a separation oracle.

Furthermore, notice that condition (1) of box-w-DI solvability is a weaker form of box-TDIness due to two reasons. First, our objective vector $w \in \{0,1\}^N$ is fixed, whereas in box-TDIness, dual integrality has to hold for any integral objective vector. Second, when dealing with box-TDIness, one can additionally add lower bounds $x \geq \ell$ on x in (4.2), still getting a linear program with an optimal integral dual solution.

For all problems we discuss here, we even have box-TDI descriptions. The only additional property needed for a box-TDI system to be box-w-DI solvable, is that one can efficiently find an optimal integral dual solution. However, such procedures are known for essentially all classical box-TDI systems. In particular, this applies to the classical polyhedral descriptions of the independent sets of a matroid or the intersection of two matroids, stable sets in bipartite graphs, s-t flows, and any problem whose constraint matrix can be chosen to be totally unimodular (TU) and of polynomial size.

Since our only access to the feasible set is via the oracle guaranteed by box-w-DI solvability, we have to be clear about what we consider to be the input size when talking about polynomial time algorithms. In addition to the binary encodings of B, c, we also assume that the binary encodings of the optimal value of (4.3) and the integral optimal vector $r \in \mathbb{Z}^N_{\geq 0}$ returned by the box-w-DI oracle are part of the input size. This implies that in particular, the binary encoding of $\nu^* = \max\{w^T x \mid Ax \leq b, x \geq 0\}$ is part of the input size.

[2]In some cases one can get around this problem by using an *extended formulation*. This is a lifting of a polytope in a higher dimension with the goal to obtain a lifted polytope with an inequality description of only polynomial size (see [67, 36]).

4.2.3 Our results

The following theorem summarizes our main result for obtaining 2-pseudoapproximations.

Theorem 58. *There is an efficient* 2*-pseudoapproximation for any interdiction problem of type* (4.1) *if the following conditions are satisfied:*

1. *The objective function* w *is a* $\{0,1\}$*-vector, i.e.,* $w \in \{0,1\}^N$*,*
2. *the description of the feasible set* $\mathcal{X} \subseteq \mathbb{R}^N$ *is* w*-down-closed, and*
3. *there is a box-*w*-DI solvable description of* $\operatorname{conv}(\mathcal{X})$*.*

Using well-known box-TDI description of classical combinatorial optimization problems (see [93]), Theorem 58 leads to 2-pseudoapproximations for the interdiction of many combinatorial optimization problems.

Corollary 59. *There is a* 2*-pseudoapproximation for interdicting maximum cardinality independent sets of a matroid or the intersection of two matroids, and maximum* s*-*t *flows. Furthermore, there is a* 2*-pseudoapproximation for all problems where a maximum cardinality set has to be found with respect to down-closed constraints captured by a TU matrix. For example, this includes maximum* b*-stable sets in bipartite graphs.*

We recall that for the maximum s-t flow problem, a 2-pseudoapproximation was already known due to Burch et al. [26].

Furthermore, for interdicting independent sets of a matroid we obtain stronger results by leveraging the strong combinatorial structure of matroids to adapt our approach. Consider a matroid $M = (N, \mathcal{I})$ on ground set N with independent sets $\mathcal{I} \subseteq 2^N$. We recall the definition of a matroid, which requires $\mathcal{I}$ to be a nonempty set such that: (i) $\mathcal{I}$ is an independence system, i.e., $I \in \mathcal{I}$ and $J \subseteq I$ implies $J \in \mathcal{I}$, and (ii) for any $I, J \in \mathcal{I}$ with $|I| < |J|$, there exists $e \in J \setminus I$ such that $I \cup \{e\} \in \mathcal{I}$. We typically assume that a matroid is given by an *independence oracle*, which is an

oracle that, for any $I \subseteq N$, returns whether $I \in \mathcal{I}$ or not. See [93, Volume B] for more information on matroids.

For matroids, we can get a 2-pseudoapproximation even for arbitrary nonnegative weight functions w, i.e., for interdicting the *maximum weight* independent set of a matroid. Furthermore, we can also handle monotone nonnegative submodular interdiction costs c. A *submodular function* c defined on a ground set N, is a function $c : 2^N \to \mathbb{R}_{\geq 0}$ that assigns a nonnegative value $c(S)$ to each set $S \subseteq N$ and fulfills the following property of *economies of scale*:

$$c(A \cup \{e\}) - c(A) \geq c(B \cup \{e\}) - c(B) \qquad A \subseteq B \subseteq N, e \in N \setminus B.$$

In words, the marginal cost of interdicting an element is lower when more elements will be interdicted. Economies of scale can often be a natural property in interdiction problems. It allows for modeling dependencies that are sometimes called *cascading failures* or *chain-reactions*, depending on the context. More precisely, it may be that the interdiction of a set of elements $S \subseteq N$ will render another element $e \in N$ unusable. This can be described by a submodular interdiction cost c which assigns a marginal cost of 0 to the element e, once all elements of S have been removed. Still, removing only e may have a strictly positive interdiction cost. Such effects cannot be captured with linear interdiction costs. A submodular function $c : 2^N \to \mathbb{R}_{\geq 0}$ is called *monotone* if $c(A) \leq c(B)$ for $A \subseteq B \subseteq N$. We typically assume that a submodular function f is given through a *value oracle*, which is an oracle that, for any set $S \subseteq N$, returns $f(S)$.

Theorem 60. *There is an efficient 2-pseudoapproximation to interdict the problem of finding a maximum weight independent set in a matroid, with monotone nonnegative submodular interdiction costs. The following is a formal description of this interdiction problem:*

$$\min_R \{\max_I \{w(I) \mid I \in \mathcal{I}, I \cap R = \emptyset\} \mid R \subseteq N, c(R) \leq B\},$$

where $c : 2^N \to \mathbb{R}_{\geq 0}$ is a monotone nonegative submodular function, and $w \in \mathbb{Z}^N_{\geq 0}$ [3]*. The matroid is given through an independence oracle and the submodular cost function c through a value oracle.*

Finally, we show that our approach can sometimes be refined to obtain stronger approximation guarantees. We illustrate this on the interdiction version of the b-stable set problem in bipartite graphs. Here, a bipartite graph $G = (V, E)$ with bipartition $V = I \cup J$, and a vector $b \in \mathbb{Z}^E_{>0}$ is given. A b-stable set in G is a vector $x \in \mathbb{Z}^V_{\geq 0}$ such that $x(i) + x(j) \leq b(\{i, j\})$ for $\{i, j\} \in E$. Hence, by choosing b to be the all-ones vector, we obtain the classical stable set problem. Because it can be formulated as a linear program with TU constraints, finding a maximum cardinality b-stable set in a bipartite graph is efficiently solvable. However, its interdiction version is easily seen to be NP-hard, by a reduction from the knapsack problem. Exploiting the adjacency properties of a polytope that is crucial in our analysis we can even get a true approximation algorithm, which does not violate the budget. More precisely, we obtain a polynomial-time approximation scheme (PTAS), which is an algorithm that, for any $\epsilon > 0$, computes efficiently an interdiction set leading to a value of at most $1 - \epsilon$ times the optimal value.

Theorem 61. *There is a PTAS for the interdiction of b-stable sets in bipartite graphs.*

We complete this discussion of interdicting b-stable sets in bipartite graphs by showing that the special case of interdicting stable sets in bipartite graph, i.e., $b = 1$, is efficiently solvable. This is done through a reduction to a polynomial number of efficiently solvable matroid intersection problems.

Theorem 62. *The problem of interdicting the maximum cardinality stable set in a bipartite graph can be solved efficiently.*

The above theorem generalizes a result by Bazgan, Toubaline and Tuza [13] who showed that interdiction of stable sets in bipartite graphs can be done efficiently when all interdiction costs are one. Our result applies to arbitrary interdiction costs.

[3] Notice that the integrality requirement for w is not restrictive. Any $w \in \mathbb{Q}^N_{\geq 0}$ can be scaled up to an integral weight vector without changing the problem.

4.3 General approach to obtain 2-pseudoapproximations

Consider an interdiction problem that fulfills the conditions of Theorem 58. As usual, let N be the ground set of our problem, $w \in \{0,1\}^N$ be the objective vector, and we denote by $\mathcal{X} \subseteq \mathbb{R}^N_{\geq 0}$ the set of feasible solutions. Furthermore, let $\{x \in \mathbb{R}^N \mid Ax \leq b, x \geq 0\} = \text{conv}(\mathcal{X})$ be a box-w-DI solvable description of $\text{conv}(\mathcal{X})$. We denote by m the number of rows of A.

One key ingredient in our approach is to model interdiction as a modification of the objective instead of a restriction of sets that can be chosen. This is possible due to w-down-closedness. More precisely, we replace the description of the interdiction problem given by (4.1) with the following min-max problem.

$$\begin{aligned} \min_{r} \max_{x} \quad & (w-r)^T x \\ Ax &\leq b \\ x &\geq 0 \\ c^T r &\leq B \\ r &\in \{0,1\}^N \end{aligned} \tag{4.4}$$

We start by showing that (4.1) and (4.4) are equivalent in the following sense. For any interdiction set $R \subseteq N$, let

$$\begin{aligned} \phi(R) :&= \max\{w^T x \mid x \in \text{conv}(\mathcal{X}), x(e) = 0 \; \forall e \in R\} \\ &= \max\{w^T x \mid Ax \leq b, x \geq 0, x(e) = 0 \; \forall e \in R\}. \end{aligned}$$

Hence, $\phi(R)$ is the value of the problem (4.1) for a fixed set R. Similarly, we define for any charac-

teristic vector $r \in \{0,1\}^N$ of an interdiction set

$$\begin{aligned}\psi(r) :&= \max\{(w-r)^T x \mid x \in \operatorname{conv}(\mathcal{X})\} \\ &= \max\{(w-r)^T x \mid Ax \leq b, x \geq 0\}.\end{aligned}$$

Thus, $\psi(r)$ is the value of (4.4) for a fixed vector $r \in \{0,1\}^N$.

Lemma 63. *For every interdiction set $R \subseteq N$, we have $\phi(R) = \psi(\chi^R)$. In particular, this implies that* (4.1) *and* (4.4) *have the same optimal value, and optimal interdiction sets R to* (4.1) *correspond to optimal characteristic vectors χ^R to* (4.4) *and vice versa.*

We show Lemma 63 based on another lemma stated below that highlights an important consequence of w-down-closedness, which we will use later again.

Lemma 64. *Let $r \in \mathbb{R}^N_{\geq 0}$ and $U = \{e \in N \mid r(e) \geq 1\}$. Then there exists $x \in \mathbb{R}^N_{\geq 0}$ with $x(e) = 0\ \forall e \in U$, such that x is an optimal solution to the following linear program.*

$$\begin{aligned}\max_x \quad & (w-r)^T x \\ & Ax \leq b \\ & x \geq 0\end{aligned} \tag{4.5}$$

Proof. Among all optimal solutions to the above linear program, let x^* be one that minimizes $x^*(U)$. Notice that x^* can be chosen to be a vertex of $\operatorname{conv}(\mathcal{X}) = \{x \in \mathbb{R}^N \mid Ax \leq b, x \geq 0\}$, since x^* can be obtained by minimizing the objective χ^U over the face of all optimal solutions to the above LP. We have to show $x^*(U) = 0$. Assume for the sake of contradiction that there is an element $e \in U$ such that $x^*(e) > 0$. Since x^* is a vertex of $\operatorname{conv}(\mathcal{X})$, we have $x^* \in \mathcal{X}$. By w-down-closedness of $\mathcal{X}$,

there is a vector $x' \in \mathcal{X}$ with $x' \leq x^*$, $x'(e) = 0$, and $w^T x' \geq w^T x^* - x^*(e)$. We thus obtain

$$\begin{aligned}(w - r)^T x' &\geq w^T x^* - x^*(e) - r^T x' \\ &\geq w^T x^* - x^*(e) - (r^T x^* - x^*(e)) \\ &= (w - r)^T x^*,\end{aligned}$$

where in the second inequality we used $r^T x' \leq r^T x^* - x^*(e)$, which follows from $x' \leq x^*$ together with $x'(e) = 0$ and $r(e) \geq 1$. Hence, x' is an optimal solution to the LP with $x'(U) < x^*(U)$, which violates the definition of x^* and thus finishes the proof. □

Proof of Lemma 63. Let $R \subseteq N$ be an interdiction set, and $r = \chi^R$ its characteristic vector. Let $x \in \mathcal{X}$ be an optimal solution to the maximization problem defining $\phi(R)$, i.e., $w^T x = \phi(R)$ and $x(e) = 0 \; \forall e \in R$. We have

$$\psi(r) \geq (w - r)^T x = w^T x = \phi(R),$$

where the first equality follows from $x(e) = 0$ for $e \in R$. Hence, $\psi(r) \geq \phi(R)$.

Conversely, let $x \in \text{conv}(\mathcal{X})$ be an optimal solution to the maximization problem defining $\psi(r)$, i.e., $\psi(r) = (w - r)^T x$. By Lemma 64, x can be chosen such that $r^T x = x(R) = 0$. Hence,

$$\psi(r) = (w - r)^T x = w^T x \leq \phi(R),$$

and thus $\phi(R) = \psi(r)$. □

Hence, (4.4) is an alternative description of the interdiction problem (4.1) in which we are

interested. In a next step we relax the integrality of r to obtain the following mathematical program.

$$\begin{aligned} \min_{r\in\mathbb{R}^n} \max_{x\in\mathbb{R}^n} \quad & (w-r)^T x \\ Ax &\leq b \\ x &\geq 0 \\ c^T r &\leq B \\ 1 \geq r &\geq 0 \end{aligned} \tag{4.6}$$

As we will show next, the constraint $1 \geq r$ can be dropped due to w-down-closedness without changing the objective. This leads to the following problem.

$$\begin{aligned} \min_{r\in\mathbb{R}^n} \max_{x\in\mathbb{R}^n} \quad & (w-r)^T x \\ Ax &\leq b \\ x &\geq 0 \\ c^T r &\leq B \\ r &\geq 0 \end{aligned} \tag{4.7}$$

The following lemma not only highlights that the objective values of (4.6) and (4.7) match, but also that any optimal interdiction vector r of (4.7) can easily be transformed to an optimal interdiction vector of (4.6). Thus, we can restrict ourselves to (4.7). We recall that $\psi(r)$ corresponds to the inner maximization problem of both (4.6) and (4.7) for a fixed vector r.

Lemma 65. *We have*

$$\psi(r) = \psi(r \wedge 1) \qquad \forall r \in \mathbb{R}^N_{\geq 0},$$

where $r \wedge 1$ is the component-wise minimum between r and the all-ones vector $1 \in \mathbb{R}^N$. This implies that (4.6) *and* (4.7) *have the same optimal value, and if r is optimal for* (4.7) *then $r \wedge 1$ is optimal for* (4.6)*.*

Proof. Let $r \in \mathbb{R}^N_{\geq 0}$ and consider the maximization problem that defines $\psi(r)$, which is the same as

the linear program desribed by (4.5). Furthermore, let $r' = r \wedge 1$, and let $U = \{e \in N \mid r(e) \geq 1\}$. In particular, r and r' are identical on $N \setminus U$. We clearly have $\psi(r') \geq \psi(r)$ by monotonicity of ψ. Therefore only $\psi(r') \leq \psi(r)$ has to be shown.

By Lemma 64, there exists an optimal vector x to the maximization problem defining $\psi(r')$ that satisfies $x(e) = 0 \; \forall e \in U$. Furthermore, by using that r and r' are identical on $N \setminus U$ we obtain

$$\begin{aligned}
\psi(r') &= (w - r')^T x \\
&= w^T x - \sum_{e \in N \setminus U} r'(e)x(e) - \sum_{e \in U} r'(e)x(e) \\
&= w^T x - \sum_{e \in N \setminus U} r(e)x(e) - \sum_{e \in U} r'(e)x(e) && (r \text{ and } r' \text{ are identical on } N \setminus U) \\
&= w^T x - \sum_{e \in N \setminus U} r(e)x(e) - \sum_{e \in U} r(e)x(e) && (x(e) = 0 \text{ for } e \in U) \\
&= (w - r)^T x \\
&\leq \psi(r),
\end{aligned}$$

as desired. □

Interestingly, problem (4.7) has already been studied in a different context. It can be interpreted as the problem to inhibit a linear optimization problem by a continuous and limited change of the objective vector w. In particular, Frederickson and Solis-Oba [43, 44] presented efficient algorithms to solve this problem when the underlying combinatorial problem is the maximum weight independent set problem in a matroid. Jüttner [66] presents efficient procedures for polymatroid intersection and minimum cost circulation problem. Also, Jüttner provides an excellent discussion how such problems can be solved efficiently using parametric search techniques.

However, our final goal is quite different from their setting since, eventually, we need to find a $\{0, 1\}$-vector r. This difference is underlined by the fact that without integrality, problem (4.7) can often be solved efficiently, whereas the interdiction problems we consider are NP-hard.

Still, we continue to further simplify (4.7) in a similar way as it was done by Jüttner [66]. For a fixed r, the inner maximization problem in (4.7) is a linear program with a finite optimum, since $\mathcal{X}$ is bounded and nonempty by assumption, and therefore also $\text{conv}(\mathcal{X}) = \{x \in \mathbb{R}^N \mid Ax \leq b, x \geq 0\}$ is bounded and nonempty. Hence, we can leverage strong duality to dualize the inner maximization into a minimization problem. We thus end up with a problem where we first minimize over r and then over the dual variables, which we can rewrite as a single minimization, thus obtaining the following LP.

$$\begin{aligned} \min \quad & b^T y & \\ & A^T y + r \;\geq\; w \\ & y \;\geq\; 0 \\ & c^T r \;\leq\; B \\ & r \;\geq\; 0 \end{aligned} \tag{4.8}$$

Hence, by strong duality, the optimal value of (4.8) is the same as the optimal value of (4.7). This reduction also shows why problem (4.7), which has no integrality constraints on r, can often be solved efficiently; This can often be achieved by obtaining an optimal vector $r \in \mathbb{R}^N_{\geq 0}$ by solving the LP (4.8) with standard linear programming techniques.

What we will do in the following is to show that there is an optimal solution (r, y) for (4.7) which can be written as a convex combination of two integral solutions (r^1, y^1) and (r^2, y^2) that may violate the budget constraint. Similar to a reasoning used in Burch et al. [26] this then implies than one of r^1 and r^2 is a 2-pseudoapproximation.

To compute r^1 and r^2, we move the constraint $c^T r \leq B$ in (4.8) into the objective via Lagrangian duality, by introducing a multiplier $\lambda \geq 0$ (see [22] for more details). We do this in two steps to highlight that the resulting Lagrangian dual problem can be solved via the oracle guaranteed by box-w-DI solvability. First, we dualize (4.8) to the obtain the following linear program, which is nicely structured in the sense that for any fixed $\lambda \geq 0$, it corresponds to optimizing a linear function

over $\text{conv}(\mathcal{X})$ with upper box constraints.

$$\begin{aligned} \max \quad & w^T z - \lambda B \\ & Az \leq b \\ & z - \lambda c \leq 0 \\ & z \geq 0 \\ & \lambda \geq 0 \end{aligned} \tag{4.9}$$

Consider the above LP as a problem parameterized by $\lambda \geq 0$. Since $\{x \in \mathbb{R}^N \mid Ax \leq b, x \geq 0\}$ is box-w-DI solvable, the LP obtained from (4.9) by fixing $\lambda \geq 0$ has an optimal integral dual solution. Furthermore, such an optimal integral dual solution can be found efficiently by box-w-DI solvability. The dual problem of (4.9) for a fixed $\lambda \geq 0$ is the problem LP(λ) below with optimal objective value $L(\lambda)$.

$$\begin{aligned} L(\lambda) = \min \quad & b^T y - \lambda(B - c^T r) \\ & A^T y + r \geq w \\ & y \geq 0 \\ & r \geq 0 \end{aligned} \tag{LP(λ)}$$

Notice that LP(λ) is indeed the problem obtained from (4.8) by moving the constraint $c^T r \leq B$ into the objective using λ as Lagrangian multiplier.

The following lemma summarizes the relationships between the different problems we introduced.

Lemma 66. *The optimal values of* (4.6), (4.7), (4.8), (4.9) *are all the same and equal to* $\max_{\lambda \geq 0} L(\lambda)$.

Furthermore, the common optimal value of the above-mentioned problems are a lower bound to OPT*, the optimal value of the considered interdiction problem* (4.1).

Proof. Problem (4.6) and (4.7) have identical optimal values due to Lemma 65. The LP (4.8) was obtained from (4.7) by dualizing the inner maximization problem. Both problems have the same optimal value due to strong duality, which holds since $\text{conv}(\mathcal{X}) = \{x \in \mathbb{R}^N \mid Ax \leq b, x \geq 0\}$ is a

nonempty polytope and thus, the inner maximization problem of (4.7) has a finite optimum value for any $r \in \mathbb{R}^n$. This also shows that the optimum value of (4.7), and hence also of (4.8) and (4.6), is finite. Problems (4.9) and (4.8) are a primal-dual pair of linear programs. For this pair of LPs, strong duality holds because (4.8), and therefore also (4.9), has a finite optimum value. Finally $\max_{\geq 0} L(\lambda)$ is the same as the optimum value of (4.8) by Lagrangian duality.

It remains to observe that the optimal value of the above problems is a lower bound to OPT. We recall that by Lemma 63, problem (4.4) is a rephrasing of the original interdiction problem (4.1), and thus also has optimal value OPT. Finally, (4.6) is obtained from (4.4) by relaxation the integrality condition on r. Thus, the optimum value of (4.6)—which is also the optimum value of (4.7), (4.8), (4.9) and $\max_{\lambda \geq 0} L(\lambda)$—is less or equal to OPT, as claimed. □

The following theorem shows that we can efficiently compute an optimal dual multiplier λ^* together with two integral vectors r^1, r^2 that are optimal solutions to $LP(\lambda^*)$, one of which will turn out to be a 2-pseudoapproximation to the considered interdiction problem (4.1).

Theorem 67. *There is an efficient algorithm to compute a maximizer λ^* of $\max_{\lambda \geq 0} L(\lambda)$, and two vectors $r^1, r^2 \in \mathbb{Z}^N_{\geq 0}$ such that:*

1. *$\exists$ integral $y^1, y^2 \in \mathbb{Z}^m$ such that both (r^1, y^1) and (r^2, y^2) are optimal solutions to $LP(\lambda^*)$.*
2. *$c^T r^1 \geq B \geq c^T r^2$.*

Before proving Theorem 67, we show that it implies our main result, Theorem 58.

Theorem 68. *Let λ^* be a maximizer of $\max_{\lambda \geq 0} L(\lambda)$, let $(r^1, y^1), (r^2, y^2)$ be two optimal solutions to $LP(\lambda^*)$ with $c^T r^1 \geq B \geq c^T r^2$, and let $\alpha > 0$. Then at least one of the following two conditions holds:*

1. *$c^T r^1 \leq (1 + \frac{1}{\alpha})B$, or*
2. *$b^T y^2 \leq (1 + \alpha)L(\lambda^*)$.*

Furthermore, if (1) *holds, then* $r^1 \wedge 1$ *is the characteristic vector of a* $(1, 1 + \frac{1}{\alpha})$*-approximation to* (4.1)*. If* (2) *holds, then* $r^2 \wedge 1$ *is the characteristic vector of a* $(1 + \alpha, 1)$*-approximation to* (4.1)*.*

Proof. Before showing that either (1) or (2) holds, we show the second part of the theorem.

Assume first that (1) holds. We recall that problem (4.1) and (4.4) are equivalent due to Lemma 63. Thus, the objective value of the interdiction problem (4.1) that corresponds to $r^1 \wedge 1$ is given by $\psi(r^1 \wedge 1)$ which, by Lemma 65, is equal to $\psi(r^1)$. Hence, to show that $r^1 \wedge 1$ is a $(1, 1 + \frac{1}{\alpha})$-approximation, it suffices to prove $\psi(r^1) \leq L(\lambda^*)$, because $L(\lambda^*) \leq \text{OPT}$ by Lemma 66.

Indeed, $\psi(r^1) \leq L(\lambda^*)$ holds due to:

$$\begin{aligned} L(\lambda^*) &= b^T y^1 - \lambda^*(B - c^T r^1) && ((r^1, y^1) \text{ is a maximizer of } LP(\lambda^*)) \\ &\geq b^T y^1 && (B \leq c^T r^1 \text{ and } \lambda^* \geq 0) \\ &\geq \psi(r^1) && (y^1 \text{ is a feasible solution to the dual of the LP defining } \psi(r^1)). \end{aligned}$$

Similarly, if (2) holds then the objective value corresponding to $r^2 \wedge 1$ is

$$\begin{aligned} \psi(r^2) &\leq b^T y^2 && (y^2 \text{ is a feasible solution to the dual of the LP defining } \psi(r^2)) \\ &\leq (1 + \alpha) L(\lambda^*) && (\text{by (2)}) \\ &\leq (1 + \alpha)\,\text{OPT} && (\text{by Lemma 66}). \end{aligned}$$

Since r^2 satisfies $c^T r^2 \leq B$, the characteristic vector $r^2 \wedge 1$ is therefore indeed a $(1 + \alpha, 1)$-approximation to (4.1).

Hence, it remains to show that at least one of (1) and (2) holds. Assume for the sake of contradiction that both do not hold. Because both (r^1, y^1) and (r^2, y^2) are maximizers of $LP(\lambda^*)$, also any convex combination of these solutions is a maximizer. In particular let $\mu = \frac{\alpha}{1+\alpha}$ and consider the maximizer (r_μ, y_μ) of $LP(\lambda^*)$, where $r_\mu = \mu r^1 + (1 - \mu) r^2$ and $y_\mu = \mu y^1 + (1 - \mu) y^2$.

We obtain

$$
\begin{aligned}
L(\lambda^*) &= b^T y_\mu - \lambda^*(B - c^T r_\mu) \\
&\geq (1-\mu) b^T y^2 - \lambda^*(B - \mu c^T r^1) && \text{(ignoring } \mu b^T y^1 \text{ and } (1-\mu)\lambda^* c^T r^2\text{, which are both } \geq 0) \\
&= \frac{1}{1+\alpha} b^T y^2 - \lambda^* \left(B - \frac{\alpha}{1+\alpha} c^T r^1 \right) && \text{(using } \mu = \frac{\alpha}{1+\alpha}\text{)} \\
&> L(\lambda^*) && \text{(using that both (1) and (2) do not hold),}
\end{aligned}
$$

thus leading to a contradiction and proving the theorem.

□

Theorem 67 together with Theorem 68 imply our main result, Theorem 58, due to the following. Theorem 67 guarantees that we can compute efficiently λ^*, r^1, r^2 as needed in Theorem 68. Then, depending whether condition (1) or (2) holds, we either return $r^1 \wedge 1$ or $r^2 \wedge 1$ as our 2-pseudoapproximation. Notice that to check whether 2 holds, we have to compute $L(\lambda^*)$. This can be done efficiently due to the fact that our description of $\text{conv}(\mathcal{X})$ is box-w-DI solvable. More precisely, as already discussed, $LP(\lambda^*)$ is the dual of (4.9) for $\lambda = \lambda^*$ whose optimal value can be computed by box-w-DI solvability. Hence, it remains to prove Theorem 67.

4.3.1 Proof of Theorem 67

First we discuss some basic properties of $L(\lambda)$. We start by observing that $L(\lambda)$ is finite for any $\lambda > 0$. This follows by the fact that $L(\lambda)$ is the optimal value of (4.9) when λ is considered fixed. More precisely, for any fixed $\lambda \geq 0$, the problem (4.9) is feasible and bounded. It is feasible because $z = 0$ is feasible since $b \geq 0$. Furthermore, it is bounded since by assumption $\text{conv}(\mathcal{X}) = \{z \in \mathbb{R}^N \mid Az \leq b, z \geq 0\}$ is a polytope. Additionally, $L(\lambda)$ has the following properties, which are true for any Lagrangian dual of a finite LP (see [22] for more details):

- $L(\lambda)$ is piecewise linear.

- Let $[\lambda_1, \lambda_2]$ be one of the linear segments of $L(\lambda)$, let $t \in (\lambda_1, \lambda_2)$, and (r_t, y_t) be an optimal solution to $L(t)$. Then, (r_t, y_t) is an optimal solution for the whole segment, i.e., for any $LP(\lambda)$ with $\lambda \in [\lambda_1, \lambda_2]$. As a consequence, the slope of the segment is $c^T r_t - B$.

Also, we recall that $L(\lambda)$ can be evaluated efficiently for any $\lambda \geq 0$; since (4.9) is box-w-DI solvable, it can be solved for any fixed $\lambda \geq 0$.

We will find an optimal multiplier $\lambda^* \geq 0$ using bisection. For this, we start by showing two key properties of $L(\lambda)$. First, we show that any optimal multiplier λ^* to $L(\lambda)$ is not larger than some upper bound with polynomial input length. Second, we show that each linear segment of $L(\lambda)$ has some minimal width, which makes it possible to reach it with a polynomial number of iterations using bisection.

We recall that

$$\nu^* = \max\{w^T x \mid Ax \leq b, x \geq 0\} = \min\{b^T y \mid A^T y \geq w, y \geq 0\}$$

is the optimal value of the nominal problem without interdiction, and that $\log(\nu^*)$ is part of the input size.

Lemma 69. *If λ^* is a maximizer of $L(\lambda)$, then $\lambda^* \leq \nu^*$. Furthermore, for every $\lambda \geq \nu^*$, $r = 0$ is an optimal solution to $LP(\lambda)$.*

Proof. Let $r = 0 \in \mathbb{Z}^N$, and y^* be a minimizer of $\min\{b^T y \mid A^T y \geq w, y \geq 0\}$. Hence, in particular, $b^T y^* = \nu^*$. We first show that for any $\lambda \geq \nu^*$, the pair (r, y^*) is a minimizer of $LP(\lambda)$. Assume for the sake of contradiction that there is some $\lambda \geq \nu^*$ such that (r, y^*) is not a minimizer of $LP(\lambda)$. Let (r', y') be a minimizer of $LP(\lambda)$ which, because the dual of $LP(\lambda)$ is box-w-DI, can be assumed to be integral. Clearly, we must have $r' \neq 0 = r$, since for $r = 0$, the vector y^* attains by definition

the smallest value in $LP(\lambda)$. Hence, we obtain

$$\begin{aligned} b^T y^* - \lambda B &> b^T y' - \lambda B + \lambda c^T r' && ((r', y') \text{ attains a smaller value than } (r, y^*) \text{ in } LP(\lambda)) \\ &\geq -\lambda B + \lambda c^T r' && (b^T y' \geq 0 \text{ since } b \geq 0 \text{ and } y' \geq 0), \end{aligned}$$

which implies

$$\nu^* > \lambda c^T r'.$$

However, this is a contradiction since $\lambda \geq \nu^*$, and $c^T r' \geq 1$ because $c \in \mathbb{Z}^N_{>0}$ and $r' \in \mathbb{Z}^N_{\geq 0}$ is nonzero. Thus, (r, y^*) is indeed a minimizer of $LP(\lambda)$ for any $\lambda \geq \nu^*$. However, since $B > 0$, this implies

$$L(\nu^*) = b^T y - \nu^* B > b^T y - \lambda B = L(\lambda) \qquad \forall \lambda > \nu^*,$$

thus implying the lemma. □

Hence, Lemma 69 implies that to find a maximizer λ^* of $L(\lambda)$, we only have to search within the interval $[0, \nu^*]$.

Lemma 70. *Each segment of the piecewise linear function $L(\lambda)$ has width at least $\frac{1}{(c(N))^2}$.*

Proof. We start by deriving a property of the kinks of $L(\lambda)$, namely that they correspond to a rational value λ whose denominator is at most $\frac{1}{c(N)}$. Later we will derive from this property that the distance between any two kinks is at least $\frac{1}{(c(N))^2}$.

Let $\overline{\lambda} > 0$ be the value of a kink of $L(\lambda)$, i.e., there is one segment of the piecewise linear function $L(\lambda)$ that ends at $\overline{\lambda}$ and one that starts at $\overline{\lambda}$. We call the segment ending at $\overline{\lambda}$ the *left segment*, and the one starting at $\overline{\lambda}$ the *right segment*. Let (r^1, y^1) be an optimal solution for all $LP(\lambda)$ where λ is within the left segment. Similarly, let (r^2, y^2) be an optimal solution for the right segment. By box-w-DI solvability, we can choose (r^i, y^i) for $i \in \{1, 2\}$ to be integral. We start by

showing that r^1 and r^2 are $\{0,1\}$-vectors, i.e, $r^1, r^2 \in \{0,1\}^N$. We can rewrite $L(\lambda)$ as follows:

$$\begin{aligned} L(\lambda) &= \min\{b^T y - \lambda(B - c^T r) \mid A^T y + r \geq w, y \geq 0, r \geq 0\} \\ &= \min_{r \geq 0} \left(-\lambda B + c^T r + \max\{(w-r)^T x \mid Ax \leq b, x \geq 0\}\right) \\ &= \min_{r \geq 0} \left(-\lambda B + c^T r + \psi(r)\right), \end{aligned} \tag{4.10}$$

where the second equality follows by dualizing the LP of the first line for a fixed $r \geq 0$, and the third equality follows by the definition of ψ. By Lemma 65, we have $\psi(r) = \psi(r \wedge 1)$, and since $c \in \mathbb{Z}_{>0}$, this implies that a minimizing r is such that $r = r \wedge 1$. Thus an integral minimizing r satisfies $r \in \{0,1\}^N$, as desired.

The slope of the left segment is $\beta_1 = -B + c^T r^1$ and the slope of the right segment is $\beta_2 = -B + c^T r^2$. Let $\alpha_1 = b^T y^1$ and $\alpha_2 = b^T y^2$. Again using (4.10) we have $\alpha_i = \psi(r^i)$ for $i \in \{1,2\}$. This implies that α_i for $i \in \{1,2\}$ is integral because $\psi(r^i)$ is defined as the optimum of an LP with integral objective vector over an integral polytope $\{x \in \mathbb{R}^N \mid Ax \leq b, x \geq 0\}$.

Because $L(\lambda)$ is concave, the slope decreases strictly at each kink, i.e., $\beta_1 > \beta_2$. Furthermore, since the left and right segment touch at $\overline{\lambda}$, we have

$$\alpha_1 + \overline{\lambda}\beta_1 = \alpha_2 + \overline{\lambda}\beta_2.$$

Because $\beta_1 > \beta_2$ and $\overline{\lambda} > 0$, this implies $\alpha_1 < \alpha_2$, and $\overline{\lambda}$ can be written as

$$\overline{\lambda} = \frac{\alpha_2 - \alpha_1}{\beta_1 - \beta_2}.$$

Notice that

$$\beta_1 - \beta_2 = c^T(r^1 - r^2) \leq \|c\|_1 = c(N),$$

where we use the fact that $r^1, r^2 \in \{0,1\}^N$ for the inequality. In summary, any kink $\overline{\lambda}$ is a rational

number $\frac{p}{q}$ with $p, q \in \mathbb{Z}_{>0}$ and $q \leq c(N)$. In particular, this implies that the first segment, which goes from $\lambda = 0$ to the first kink, has width at least $\frac{1}{c(N)} \geq \frac{1}{(c(N))^2}$. The last segment clearly has infinite width. Any other segment is bordered by two kinks $\lambda_1 = \frac{p_1}{q_1}$ and $\lambda_2 = \frac{p_2}{q_2}$ with $\lambda_1 < \lambda_2$ and has therefore a width of

$$\begin{aligned}
\lambda_1 - \lambda_2 &= \frac{p_1 q_2 - p_2 q_1}{q_1 q_2} \\
&\geq \frac{1}{q_1 q_2} && \text{(since } \lambda_1 - \lambda_2 > 0\text{)} \\
&\geq \frac{1}{(c(N))^2} && \text{(since } q_1, q_2 \leq c(N)\text{)}.
\end{aligned}$$

□

We use the bisection procedure Algorithm 10 to compute λ^*, r^1, and r^2 as claimed by Theorem 67. Notice that $L(\lambda^1)$ and $L(\lambda^2)$, as needed by Algorithm 10 to determine λ^*, can be computed due to box-w-DI solvability. Algorithm 10 is clearly efficient; it remains to show its correctness.

Algorithm 10 Computing λ^*, r^1 and r^2 as claimed by Theorem 67

Initialize: $\lambda^1 = 0$, $\lambda^2 = \nu^*$, $r^1 = \chi^N$, $r^2 = 0$
for $i = 1, \ldots, 1 + \lfloor \log_2(\nu^*(c(N))^2) \rfloor$ **do**
  $\lambda = \frac{1}{2}(\lambda^1 + \lambda^2)$
  Use box-w-DI solvability oracle to compute integral $r \in \mathbb{Z}_{\geq 0}$ satisfying that there is a y such that (r, y) is an optimal solution to $LP(\lambda)$
  if $-B + c^T r \geq 0$ **then**
    $\lambda^1 = \lambda$
    $r^1 = r$
  else
    $\lambda^2 = \lambda$
    $r^2 = r$
  end if
end for
Compute λ^* as the intersection of the two segments at λ^1 and λ^2:

$$\lambda^* = \frac{L(\lambda^2) - L(\lambda^1) - \lambda^2(-B + c^T r^2) + \lambda^1(-B + c^T r^1)}{c^T(r^1 - r^2)}.$$

return λ^*, r^1, r^2.

Lemma 71. *λ^*, r^1 and r^2 as returned by Algorithm 10 fulfill the properties required by Theorem 67.*

Proof. Notice that throughout the algorithm, the following invariant is maintained: r^i is an optimal solution to $LP(\lambda^i)$ for $i \in \{1, 2\}$. Furthermore, $-B + c^T r^1 \geq 0$ and $-B + c^T r^2 < 0$. We highlight that after initialization, these two invariants are maintained because $-B + c^T \chi^N = -B + c(N) > 0$ because we assumed $B < c(N)$ to avoid the trivial special case when everything is interdicted. Additionally, $-B + c^T 0 = -B < 0$. Also note that $r^2 = 0$ is an optimal solution to $LP(\nu^*)$ by Lemma 69.

Due to this invariant and the fact that $L(\lambda)$ is concave, we know that there is a maximizer λ^* of $L(\lambda)$ within $[\lambda^1, \lambda^2)$.

Observe that the distance $\lambda^2 - \lambda^1$ halves at every iteration of the for loop. Consider now λ^1 and λ^2 after the for loop. Their distance is bounded by

$$\lambda^2 - \lambda^1 = \nu^* \left(\frac{1}{2}\right)^{1+\lfloor \log_2(\nu^*(c(N))^2) \rfloor} < \nu^* \left(\frac{1}{2}\right)^{\log_2(\nu^*(c(N))^2)} = \frac{1}{(c(N))^2}.$$

Hence, the distance between λ^2 and λ^1 is less then the width of any segment of the piecewise linear function $L(\lambda)$, due to Lemma 70. This leaves the following options. Either one of λ^1 or λ^2 is a maximizer of $L(\lambda)$, and the other one is in the interior of the segment to the left or right, respectively. Or, neither λ^1 nor λ^2 is a maximizer of $L(\lambda)$. In this case λ^1 and λ^2 are in the interior of the segment to the left and right, respectively, of the unique maximizer λ^*. In all of these cases, the solutions r^1 and r^2 are both optimal with respect to some maximizer λ^* of $L(\lambda)$, since they are on two segments that meet on an optimal multiplier λ^*.

It remains to prove that the returned λ^* is correct. Since both r^1 and r^2 are optimal solutions to $L(\lambda^*)$ for some maximizer λ^*, we have

$$L(\lambda^*) = b^T y^1 - \lambda^*(B - c^T r^1) = b^T y^2 - \lambda^*(B - c^T r^2). \tag{4.11}$$

Furthermore,

$$L(\lambda^1) = b^T y^1 - \lambda^1(B - c^T r^1), \text{ and}$$
$$L(\lambda^2) = b^T y^2 - \lambda^2(B - c^T r^2).$$

By replacing $b^T y^i = L(\lambda^i) + \lambda^i(B - c^T r^i)$ for $i \in \{1, 2\}$ in (4.11) and solving for λ^*, we obtain

$$\lambda^* = \frac{L(\lambda^2) - L(\lambda^1) - \lambda^2(-B + c^T r^2) + \lambda^1(-B + c^T r^1)}{c^T(r^1 - r^2)},$$

thus showing that the returned λ^* is indeed optimal. □

Hence, even the somewhat limited access through box-w-DI solvability that we assume to our optimization problem is enough to obtain an efficient 2-pseudoapproximation for the interdiction problem, due to the efficiency of the bisection method described in Algorithm 10. However, in many concrete settings, more efficient methods can be employed to get an optimal multiplier λ^* and optimal integral dual solutions r^1, r^2. In particular, often one can even obtain strongly polynomial procedure by employing Megiddo's parametric search technique [81]. We refer the interested reader to [66] for a technical details of how this can be done in a very similar context.

4.4 Matroids: weighted case and submodular costs

In this section we consider the problem of interdicting a feasible set $\mathcal{X} \subseteq \{0, 1\}^N$ that corresponds to the independent sets of a matroid. It turns out that we can exploit structural properties of matroids to solve natural generalization of the interdiction problem considered in Theorem 58. In particular, even for arbitrary nonnegative weight functions $w \in \mathbb{Z}^N_{\geq 0}$, we can obtain a 2-pseudoapproximation for the corresponding interdiction problem. What's more is that we can achieve this when the interdiction costs are submodular, rather than just linear.

For clarity, we first discuss in Section 4.4.1 a technique to reduce arbitrary nonnegative

weights to the case of $\{0,1\}$-objectives that was mentioned previously. In Section 4.4.2, we then build up and extend this technique to also deal with submodular interdiction costs.

4.4.1 Weighted case

Let $M = (N, \mathcal{I})$ be a matroid, and let $w : N \to \mathbb{Z}_{\geq 0}$. The canonical problem we want to interdict is the problem of finding a maximum weight independent set, i.e., $\max\{w(I) \mid I \in \mathcal{I}\}$. Let $r_w : 2^N \to \mathbb{Z}_{\geq 0}$ be the *weighted rank* function, i.e.,

$$r_w(S) = \max\{w(I) \mid I \subseteq S, I \in \mathcal{I}\}.$$

In words, $r_w(S)$ is the weight of a heaviest independent set that is contained in S. We recall a basic fact on weighted rank functions [93, Section 44.1a].

One key observation we exploit is that the maximum weight independent set can be rephrased as maximizing an all-ones objective function over the following polymatroid:

$$P_w = \{x \in \mathbb{R}^N_{\geq 0} \mid x(S) \leq r_w(S) \ \ \forall S \subseteq N\}. \tag{4.12}$$

Even more importantly, we do not only have $\max\{x(N) \mid x \in P_w\} = \max\{w(I) \mid I \in \mathcal{I}\}$, but we also have that the problem of interdicting the maximum weight independent set problem of a matroid maps to the problem of interdicting the corresponding all-ones maximization problem on the polymatroid. This is formalized through the lemma below.

Lemma 72. *For any $R \subseteq N$, we have*

$$\max\{x(N) \mid x \in P_w, x(R) = 0\} = \max\{w(I) \mid I \in \mathcal{I}, I \subseteq N \setminus R\}.$$

Proof. Observe that the right-hand side of the above equality is, by definition, equal to $r_w(N \setminus R)$.

lhs $\leq$ rhs: Let x^* be a maximizer of $\max\{x(N) \mid x \in P_w, x(R) = 0\}$. We have

$$\begin{aligned} x^*(N) &= x^*(N \setminus R) && (x^*(R) = 0) \\ &\leq r_w(N \setminus R) && (\text{since } x^* \in P_w), \end{aligned}$$

thus showing the desired inequality.

lhs $\geq$ rhs: Conversely, let I^* be a maximizer of $\max\{w(I) \mid I \in \mathcal{I}, I \subseteq N \setminus R\}$. Hence, $w(I^*) = r_w(N \setminus R)$. Define $y \in \mathbb{R}^N_{\geq 0}$ by

$$y(e) = \begin{cases} w(e) & \text{if } e \in I^* \\ 0 & \text{if } e \in N \setminus I^*. \end{cases}$$

Clearly, $y(N) = w(I^*) = r_w(N \setminus R)$. Thus, to show that the left-hand side of the equality of Lemma 72 is at least as large as the right-hand side, it suffices to show that y is feasible to the maximization problem on the left-hand side, i.e., $y(R) = 0$ and $y \in P_w$. We have $y(R) = 0$ since $I^* \subseteq N \setminus R$. Furthermore,

$$y(S) = w(S \cap I^*) \leq r_w(S) \qquad \forall S \subseteq N,$$

where the inequality follows from $S \cap I^* \in \mathcal{I}$. Hence, this implies $y \in P_w$ and completes the proof. $\square$

We therefore can focus on the problem $\max\{x(N) \mid x \in P_w,\ x(R) = 0\}$ to which we can now apply Theorem 58. For this it remains to observe that P_w is 1-down-closed because it is down-closed. Furthermore, the description of P_w given by (4.12) is box-1-DI solvable since it is well-known to be even box-TDI, a property that holds for all polymatroids [93, Section 44.3]. Furthermore, one can efficiently find an optimal integral dual solution to the problem of finding a maximum size point over (4.12) with upper box constraints. In fact, this problem can be interpreted as a maximum

cardinality polymatroid intersection problem, one polymatroid being P_w and the other one being defined by the upper box constraints. An optimal integral dual solution to the maximum cardinality polymatroid intersection problem can be found in strongly polynomial time by standard techniques (for clarity we provide some more details about this in Section 4.4.2). In summary, our technique presented in Section 4.3 to obtain 2-pseudoapproximations therefore indeed applies to this setting.

4.4.2 Submodular costs

In this section, we show how to obtain a 2-pseudoapproximation for the interdiction of the maximum weight independent set of a matroid with submodular interdiction costs. When dealing with submodular interdiction costs, we assume that the interdiction costs κ are a nonnegative and monotone submodular function $\kappa : 2^N \to \mathbb{R}_{\geq 0}$. As before, a removal set $R \subseteq N$ has to satisfy the budget constraint, i.e., $\kappa(R) \leq B$. We assume that the submodular function κ is given by a value oracle.

To design a 2-pseudoapproximation, we will describe a way to formulate the problem such that it can be attacked with essentially the same techniques as described in Section 4.3. For simplicity of presentation, and to avoid replicating reasonings introduced in Section 4.3, we focus on the key differences in this section, and refer to Section 4.3 for proofs that are essentially identical.

We extend the model for the weighted case. A variable $q(S)$ is introduced for each set $S \subseteq N$. In the non-relaxed mathematical program, we have $q \in \{0,1\}^{2^N}$, and only one variable $q(S)$ is equal to one, which indicates the set S of elements we interdict. Below is a mathematical description of a relaxation, where we allow the variables $q(S)$ to take real values. If instead of allowing $q(S) \in \mathbb{R}_{\geq 0}$, we set $q(S) \in \{0,1\}$, then the mathematical program below would be an exact

description of the interdiction problem with submodular interdiction costs.

$$\begin{aligned} \min_{q\in\mathbb{R}^{2^N}} \max_{x\in\mathbb{R}^n} \quad & (1-\textstyle\sum_{S\subseteq N}\chi^S\cdot q(S))^T x \\ & x(S) \;\leq\; r_w(S) \quad \forall S\subseteq N \\ & x \;\geq\; 0 \\ & \textstyle\sum_{S\subseteq N}\kappa(S)\cdot q(S) \;\leq\; B \\ & \textstyle\sum_{S\subseteq N} q(S) \;\leq\; 1 \\ & q(S) \;\geq\; 0 \qquad \forall S\subseteq N \end{aligned} \tag{4.13}$$

We start by dropping the constraint $\sum_{S\subseteq N} q(S) \leq 1$. As we will see later, this does not change the objective value. This step is similar to dropping the constraint $r \leq 1$ when going from (4.6) to (4.7) in the standard setting of our framework without submodular interdiction costs. We thus obtain the following mathematical program.

$$\begin{aligned} \min_{q\in\mathbb{R}^{2^N}} \max_{x\in\mathbb{R}^n} \quad & (1-\textstyle\sum_{S\subseteq N}\chi^S\cdot q(S))^T x \\ & x(S) \;\leq\; r_w(S) \quad \forall S\subseteq N \\ & x \;\geq\; 0 \\ & \textstyle\sum_{S\subseteq N}\kappa(S)\cdot q(S) \;\leq\; B \\ & q(S) \;\geq\; 0 \qquad \forall S\subseteq N \end{aligned} \tag{4.14}$$

Now, by dualizing the inner problem we get the following LP.

$$\begin{aligned} \min \quad & \textstyle\sum_{S\subseteq N} r_w(S)y(S) \\ & \left(\textstyle\sum_{S\subseteq N:e\in S} y(S)\right) + \left(\textstyle\sum_{S\subseteq N:e\in S} q(S)\right) \;\geq\; 1 \quad \forall e\in N \\ & y(S) \;\geq\; 0 \quad \forall S\subseteq N \\ & q(S) \;\geq\; 0 \quad \forall S\subseteq N \\ & \textstyle\sum_{S\subseteq N}\kappa(S)\cdot q(S) \;\leq\; B \end{aligned} \tag{4.15}$$

As in the case with linear interdiction costs, we dualize the budget constraint with a

Lagrangian multiplier λ to obtain the following family of LPs, parameterized by λ:

$$\begin{aligned} L(\lambda) = \min \quad & \textstyle\sum_{S\subseteq N} r_w(S)y(S) + \lambda\left(\sum_{S\subseteq N}\kappa(S)\cdot q(S)\right) - \lambda B \\ & \left(\textstyle\sum_{S\subseteq N:e\in S} y(S)\right) + \left(\sum_{S\subseteq N:e\in S} q(S)\right) \;\geq\; 1 \quad \forall e \in N \\ & y(S) \;\geq\; 0 \quad \forall S \subseteq N \\ & q(S) \;\geq\; 0 \quad \forall S \subseteq N \end{aligned} \tag{LP(λ))}$$

It remains to observe that for any $\lambda \geq 0$, LP(λ) is the dual of a maximum cardinality polymatroid intersection problem—when forgetting about the constant term $-\lambda B$—where the two polymatroids are defined by the submodular functions r_w and $\lambda \cdot \kappa$, respectively. A key result in this context is that there is a set $A \subseteq N$ such that the optimal primal value, which is equal to the optimal dual value by strong duality, is equal to $\lambda\kappa(A) + r_w(N \setminus A)$ (see [93, Section 46.2]). This implies that defining $q(A) = 1$, $y(N \setminus A) = 1$, and setting all other entries of q and y to zero is an optimal solution to (LP(λ)). Furthermore, such a set A can be found in strongly polynomial time [93, Section 47.1]. Note that this fact also implies that dropping the constraint $\sum_{S\subseteq N} q(S) \leq 1$ when going from (4.13) to (4.14) did not change the objective value of the mathematical program. Furthermore, we can evaluate $L(\lambda)$ efficiently for any $\lambda \geq 0$.

From this point on, the approach is identical to the one presented in Section 4.3 for linear interdiction costs. More precisely, we determine the optimal dual multiplier λ^* and two optimal dual solutions (q^1, y^1), (q^2, y^2) to $LP(\lambda^*)$ such that

1. The dual solutions have the above-mentioned property that all four vectors y^1, q^1, y^2, q^2 only have 0-entries with the exception of a single 1-entry. Let $R_1, R_2 \subseteq N$ be the sets such that $q^1(R^1) = q^2(R^2) = 1$.
2. One solution has interdiction cost that is upper bounded by the budget and one has an interdiction cost that is lower bounded by the budget, i.e., $\kappa(A^1) \leq B \leq \kappa(A^2)$.

The value λ^* and vectors q^1, y^1, q^2, y^2 can either be found by bisection, as described in Section 4.3, or they can be obtained in strongly polynomial time via Megiddo's parametric search technique (see [66] for details). An identical reasoning as used in Theorem 68 shows that one of R^1 or R^2 is a 2-pseudoapproximation.

4.5 Refinements for bipartite b-stable set interdiction

This section specializes our approach to the interdiction of b-stable sets in a bipartite graph. We recall that given is a bipartite graph $G = (V, E)$ with bipartition $V = I \cup J$ and edge capacities $b \in \mathbb{Z}^E_{\geq 0}$. A b-stable set is a vector $x \in \mathbb{Z}^N_{\geq 0}$ such that $x(i) + x(j) \leq b(\{i, j\})$ for each $\{i, j\} \in E$. The *value* of a b-stable set x is given by $x(V)$. The maximum b-stable set problem asks to find a b-stable set of maximum value. Furthermore, we are given an interdiction cost $c : V \to \mathbb{Z}_{>0}$ for each vertex, and an interdiction budget $B \in \mathbb{Z}_{>0}$. As usual, the task is to remove a subset $R \subseteq V$ with $c(R) \leq B$ such that value of a maximum b-stable set in the graph obtained from G by removing R is as small as possible.

In Section 4.5.1 we show how our approach can be adapted to get a PTAS for b-stable set interdiction, thus proving Theorem 61. In Section 4.5.2 we complete the discussion on b-stable set interdiction by presenting an exact algorithm to solve the interdiction problem of the classical stable set problem in bipartite graphs, which corresponds to the case when b is the all-ones vector.

Before presenting these results, we remark that b-stable set problem has also a well-known vertex-capacitated variant. In this case an additional vector $u \in \mathbb{Z}^V_{\geq 0}$ is given and constraints $x \leq u$ are imposed. The vertex-capacitated problem can easily be reduced to the uncapacitated problem by adding two additional vertices v_I, v_J, where v_I is added to I and v_J to J, and connecting v_I to all vertices in J and v_J to all vertices in I. Finally, by choosing $b(\{v_I, j\}) = u(j)$ for $j \in J$ and $b(\{v_J, i\}) = u(i)$ for $i \in I$, one obtains a b-stable set problem that is equivalent to the vertex-capacitated version. Furthermore, a vertex interdiction strategy for minimizing the maximum b-

independent set problem in this auxiliary graph carries over exactly to the vertex-capacitated variant. Thus, the approach we present can also deal with vertex capacities.

4.5.1 PTAS BY EXPLOITING ADJACENCY STRUCTURE

As in our general approach, we start with the relaxation (4.6). Below, we adapt the description of the relaxation to this specialized setting highlight some structural aspects of the problem.

$$\begin{aligned} \min_{r\in\mathbb{R}^V} \max_{x\in\mathbb{R}^V} \quad & (1-r)^T x & \\ & Ax \;\leq\; b & \\ & x \;\geq\; 0 & \\ & c^T r \;\leq\; B & \\ & 0 \;\leq\; r \;\geq\; 1 & \end{aligned} \tag{4.16}$$

Notice that the matrix $A \in \{0,1\}^{E\times V}$ is the incidence matrix of the bipartite graph G, i.e., $A(e,v) = 1$ if and only if $v \in V$ is one of the endpoints of $e \in E$. This matrix is well known to be totally unimodular (TU) [73]. Similar to our general approach, we could now drop the constraint $r \leq 1$. However, since this does not lead to a further simplification in this setting, we will keep this constraint. Following our general approach, we dualize the inner maximization problem to obtain the following linear program.

$$\begin{aligned} \min \quad & b^T y & \\ & A^T y \;+ r \;\geq\; 1 & \\ & y \qquad\quad \geq\; 0 & \\ & c^T r \;\leq\; B & \\ & 0 \;\leq\; r \;\leq\; 1 & \end{aligned} \tag{4.17}$$

Observe that $\{0,1\}$-solutions to (4.17) have a nice combinatorial interpretation. More precisely, they correspond to a subset $R \subseteq V$ of the vertices (where $\chi^R = r$) with $c(R) \leq B$ and an edge set $F \subseteq E$ (where $\chi^F = y$) such that F is an edge cover in the graph obtained from G by

removing the vertices R.

Not surprisingly, apart from the budget constraint $c^T r \leq B$, the feasible region of the above LP closely resembles the bipartite edge cover polytope. We will make this link more explicit in the following with the goal to exploit well-known adjacency properties of the bipartite edge cover polytope. First, notice that for any feasible solution (y, r) to (4.8), the vector $(y \wedge 1, r)$ is also feasible with equal or lower objective value. This follows from the fact that A is a $\{0, 1\}$-matrix. Hence, we can add the constraint $y \leq 1$ without changing the problem to obtain the following LP.

$$\begin{aligned} \min \quad & b^T y & & \\ & A^T y + r & \geq & \ 1 \\ & c^T r & \leq & \ B \\ & 0 \leq y & \leq & \ 1 \\ & 0 \leq r & \leq & \ 1 \end{aligned} \tag{4.18}$$

The feasible region of the above LP is given by intersection the polyope

$$P = \left\{ \begin{pmatrix} y \\ r \end{pmatrix} \in \mathbb{R}^{|E|+|V|} \;\middle|\; A^T y + r \geq 1, 0 \leq y \leq 1, 0 \leq r \leq 1 \right\}$$

with the half-space $\{(y, r) \in \mathbb{R}^{|E|+|V|} \mid c^T r \leq B\}$. Notice that P is integral because the matrix A is TU. The key property we exploit is that P has very well-structured adjacency properties, because it can be interpreted as a face of a bipartite edge cover polytope, a polytope whose adjacency structure is well known. More precisely, it turns out that any two adjacent vertices of P represent solutions that do not differ much in terms of cost and objective function. Hence, similar to our general approach, we compute two vertex solutions of P, one over budget but with a good objective value and the other one under budget, with the additional property that they are adjacent on P. We then return the one solution that is budget-feasible. This procedure as stated does not yet lead to a PTAS, but it can be transformed into one by a classical preprocessing technique that we will briefly

mention at the end.

We start by introducing a bipartite edge cover polytope P' such that P is a face of P'. To simplify the exposition, we do a slight change to the above sketch of the algorithm. More precisely, we will restate (4.18) in terms of a problem on P' and then work on the polytope P' instead of P. We will define P' with a system of linear constraints. It has two new rows and one new variable r_{IJ} in addition to the constraints $A^T y + r \geq 1$ of P. The rows correspond to two new vertices in the graph, one in I and one in J, and the new variable is for an edge between the two new vertices. The updated constraints are

$$\underbrace{\begin{pmatrix} A^T & I & 0 \\ 0 & (\chi^J)^T & 1 \\ 0 & (\chi^I)^T & 1 \end{pmatrix}}_{D:=} \begin{pmatrix} y \\ r \\ r_{\mathrm{IJ}} \end{pmatrix} \geq 1, \tag{4.19}$$

where $\chi^I, \chi^J \in \{0,1\}^V$ are the characteristic vectors of $I \subseteq V$ and $J \subseteq V$, respectively. Let D be the $\{0,1\}$-matrix on the left-hand side of the constraint (4.19). Notice that D is the vertex-edge incidence matrix of a bipartite graph $G' = (V', E')$, where G' is obtained from G as follows: add one new vertex w_I to I and one new vertex w_J to J; then connect w_I to all vertices in $J \cup \{w_J\}$ and w_J to all vertices in I. Hence, $I' = I \cup \{w_I\}$ and $J' = J \cup \{w_J\}$ is a bipartition of V'. Since D is an incidence matrix of a bipartite graph, it is TU. For easier reference to the different types of edges in G' we partition E' into the edge E, the edge set

$$E_R = \{\{w_I, j\} \mid j \in J\} \cup \{\{i, w_J\} \mid i \in I\},$$

and the single edge $f = \{w_I, w_J\}$, i.e., $E' = E \cup E_R \cup \{f\}$.

Now consider the edge cover polytope that corresponds to D:

$$P' = \left\{ \begin{pmatrix} y \\ r \\ r_{IJ} \end{pmatrix} \in \mathbb{R}^{|E|+|V|+1}) \;\middle|\; D \cdot \begin{pmatrix} y \\ r \\ r_{IJ} \end{pmatrix} \geq 1, 0 \leq y, r, r_{IJ} \leq 1 \right\}.$$

Notice that P is obtained from P' by considering the face of P' defined by $r_{IJ} = 1$, and projecting out the variable r_{IJ}. Every vertex y, r, r_{IJ} of P' is a characteristic vector of an edge cover in G', where y represents the characteristic vector of the edges in E, the vector r is the characteristic vector of the edges in E_R, and $r_{IJ} = 1$ indicates that f is part of the edge cover.

We can now restate (4.18) as follows in terms of P':

$$\min \left\{ b^T y \;\middle|\; \begin{pmatrix} y \\ r \\ r_{IJ} \end{pmatrix} \in P', c^T r \leq B \right\}. \tag{4.20}$$

Indeed, one can always choose for free $r_{IJ} = 1$ in the above LP, since r_{IJ} does not appear in the objective. Furthermore, when setting $r_{IJ} = 1$, the LP (4.20) has the same feasible vectors (y, r) as (4.18). We can thus focus on (4.20) instead of (4.18).

One can interpret an edge cover F in G' as an interdiction strategy of the original problem as follows. Every vertex $v \in V$ that is incident with either w_I or w_J through an edge of F will be interdicted. To obtain a better combinatorial interpretation of 4.20, we extend the vectors b and c to all edges E'. More precisely, b is only defined for edges in E. We set $b(e) = 0$ for $e \in E' \setminus E$. Furthermore, the vector c can be interpreted as a vector on the edges E_R, where $c(\{w_I, j\}) := c(j)$ and $c(\{i, w_J\}) := c(i)$ for $i \in I$ and $j \in J$. For $e \in E' \cup \{f\}$ we set $c(e) = 0$. Using this notation, the best $\{0, 1\}$-solution to (4.20) can be interpreted as an edge cover F of G' that minimizes $b(F)$ under the constraint $c(F) \leq B$. One can observe that the best $\{0, 1\}$-solution to (4.20) corresponds to an optimal interdiction set for the original non-relaxed interdiction problem.

Also, we want to highlight that problems of this type, where a combinatorial optimization problem has to be solved under an additional linear packing constraint with nonnegative coefficients are also known as *budgeted optimization problems* or *restricted optimization problems* and have been studied for various problem settings, like spanning trees, matchings, and shortest paths (see [48] and references therein for more details). The way we adapt our procedure to exploit adjacency properties of the edge cover polytope is inspired by procedures to find budgeted matchings and spanning trees [89, 15, 48].

We compute an optimal vertex solution $p^* = (y^*, r^*, r^*_{IJ} = 1)$ to (4.20) via standard linear programming techniques. If r^* is integral, i.e., $r^* \in \{0,1\}^V$, then r^* corresponds to an optimal interdiction set since it is optimal for the relaxation and integral. Hence, assume r^* not to be integral from now on. This implies that p^* is in the interior of an edge of P', since it is a vertex of the polytope obtained by intersecting P' with a single additional constraint. This edge of the polytope P' is described by looking at the constraints of P' that are tight with respect to the optimal vertex solution. From this description of the edge, we can efficiently compute its two endpoints $y^1, r^1, r^1_{IJ} = 1$ and $y^2, r^2, r^2_{IJ} = 1$, which are vertices of P' and therefore integral. These two solutions correspond to edge covers $F^1, F^2 \subseteq E'$ in G' with $f \in F^1 \cap F^2$. For simplicity, we continue to work with these edge covers F^1 and F^2. One of these edge covers will violate the budget constraint and be superoptimal, say the first one, i.e., $c(F^1) > B$ and $b(F^1) < b^T y^*$, and the other one strictly satisfies the budget constraint and is suboptimal, i.e., $c(F^2) < B$ and $b(F^2) > b^T y^*$. Hence, this is just a particular way to obtain two solutions as required by our general approach, with the additional property that they are adjacent on the polytope P'.

The key observation is that F^2 is not just budget-feasible, but almost optimal. We prove this by exploiting the following adjacency property of edge cover polytopes shown by Hurkens.

Lemma 73 (Hurkens [54])**.** *Two edge covers U_1 and U_2 of a bipartite graph are adjacent if and only if $U_1 \Delta U_2$ is an alternating cycle or an alternating path with endpoints in $V(U_1 \cap U_2)$, where $V(U_1 \cap U_2)$ denotes all endpoints of the edges in $U_1 \cap U_2$.*

Lemma 74. *$b(F^2) \leq b^T y^* + 2b_{\max}$, where $b_{\max} = \max_{e \in E} b(e)$.*

Proof. We will prove the statement by constructing a new edge cover $Z \subseteq E'$ of G' with the following two properties:

1. $c(Z) \leq c(F^2)$, and
2. $b(Z) \leq b(F^1) + 2b_{\max}$.

We claim that this implies the result due to the following. First observe that there can be no edge cover W of G' such that $c(W) \leq c(F_2)$ and $b(W) < b(F_2)$. If such an edge cover existed, then p^* would not be an optimal solution to (4.20), because p^* is a convex combination of χ^{F^1} and χ^{F^2}, and by replacing F^2 by W one would obtain a new budget-feasible solution with lower objective value. Hence, if (1) then $b(Z) \geq b(F^2)$, which in turn implies

$$b(F^2) \leq b(Z) \overset{(2)}{\leq} b(F^1) + 2b_{\max} \leq b^T y^* + 2b_{\max}.$$

Hence, it remains to prove the existence of an edge cover $Z \subseteq E'$ satisfying (1) and (2).

By Lemma 73, $U = F^1 \Delta F^2$ is either an alternating path or cycle. In both cases, U contains at most 4 edges of E_R, at most 2 in $E_R \cap F^1$ and at most 2 in $E_R \cap F^2$. Let $E_R^1 = E_R \cap U \cap F^1$ be the up to two edges of U in $E_R \cap F^1$. Consider $X = F^1 \setminus E_R^1$. X is not necessarily an edge cover because we removed up to two edges of E_R. Hence, there may be up to 4 vertices not covered by X. However, the up to two edges of E_R that we removed to obtain X from F^1 are both incident with one of the two vertices w_I and w_J. Since $f \in X$ because $f \in F^1$, the two vertices w_I, w_J remain covered by X. Hence, there are at most two vertices $i, j \in V$ that are not covered by X. These two vertices are covered by the edge cover F^2. Thus, there are up to two edges $g, h \in F^2 \setminus F^1$ that touch i and j. Now consider the edge cover $Z = X \cup \{g, h\}$. Observe that $Z \cap E_R = (F_1 \cap F_2 \cap E_R)$. Hence, $c(Z) \leq c(F_2)$ and condition (1) holds. Furthermore, $X \subseteq F^1$, and thus $b(Z) \leq b(F^1) + b(g) + b(h) \leq b(F^1) + 2b_{\max}$, implying (2) and finishing the proof. □

Hence, F^2 corresponds to an interdiction strategy that is optimal up to $2b_{\max}$. From here, it is not hard to obtain a PTAS. Let $\epsilon > 0$. If $2b_{\max} \leq \epsilon b^T y^*$, then F^2 corresponds to an interdiction strategy that is an $(1-\epsilon)$-approximation. Otherwise, we use the following well-known guessing technique (see [89, 48]). Consider an optimal integral solution $\overline{y}, \overline{r}, \overline{r}_{IJ}$ of (4.20). The vector $\overline{r}$ of such a solution is the characteristic vector of an optimal interdiction set, and $\text{OPT} = b^T\overline{y}$ is the optimal value of our interdiction problem. We guess the $\lceil \frac{2}{\epsilon} \rceil$ heaviest edges W of $\{e \in E \mid \overline{y}(e) = 1\}$, i.e., the ones with highest b-values. This can be done by going through all subsets of E of size $\lceil \frac{2}{\epsilon} \rceil$, which is a polynomial number of subsets for a fixed $\epsilon > 0$. For each such guess we consider the resulting residual version of problem (4.20), where we set $y(e) = 1$ for each guessed edge and remove all edges of strictly higher b-values than the lowest b-value of the guessed edges. Hence, we end up with a residual problem where $b_{\max}$ is less than or equal to the b-value of any guessed edge. For the right guess W, we have $b(W) \leq \text{OPT}$ and thus get

$$b_{\max} \leq \frac{\epsilon}{2} b(W) \leq \frac{\epsilon}{2} \text{OPT},$$

implying that the set F^2 for the right guess is indeed a $(1-\epsilon)$-approximation.

Notice that if $b_{\max}$ is sufficiently small with respect to $b^T y^*$, i.e., $2b_{\max} \leq \epsilon b^T y^*$, then the expensive guessing step can be skipped.

4.5.2 Efficient algorithm for stable set interdiction in bipartite graphs

We complete the discussion on bipartite b-stable set interdiction by showing that the problem of interdicting stable sets, which are the same as 1-stable sets, in a bipartite graph can be solved in polynomial time.

We reuse the notation of the previous section. Hence, $G = (V, E)$ is a bipartite graph with bipartition $V = I \cup J$, $c : E \to \mathbb{Z}_{>0}$ are the interdiction costs, and $B \in \mathbb{Z}_{>0}$ is the interdiction budget. Furthermore, we denote by $\alpha(G)$ the size of a maximum cardinality stable set in G and by

$\nu(G)$ the size of a maximum cardinality matching. It is well-known from König's Theorem that for any bipartite graph $G = (V, E)$,

$$\alpha(G) = |V| - \nu(G).$$

Hence, the objective value of some interdiction set $R \subseteq V$ with $c(R) \leq B$ is equal to

$$\alpha(G[V \setminus R]) = |V| - |R| - \nu(G[V \setminus R]),$$

where $G[W]$ for any $W \subseteq V$ is the induced subgraph of G over the vertices W, i.e., the graph obtained from G by removing $V \setminus W$.

We start by discussing some structural properties that can be assumed to hold for at least one optimal solution. Let R^* be an optimal solution to the interdiction problem, and let $M^* \subseteq E$ be a maximum cardinality matching in $G[V \setminus R]$. By the above discussion, the value of the interdiction set R^* is

$$\alpha(G[V \setminus R^*]) = |V| - |R^*| - |M^*|. \tag{4.21}$$

In the following, we will focus on finding an optimal matching M^*, and then derive R^* from this matching. We start with a lemma that shows how R^* can be obtained from M^*. For this we need some additional notation. We number the vertices $V = \{v_1, \ldots, v_n\}$ such that $c(v_1) \leq c(v_2) \leq \cdots \leq c(v_n)$. For $\ell \in \{0, \ldots, n\}$ let $V_\ell = \{v_1, \ldots, v_\ell\}$ with $V_0 = \emptyset$. Furthermore, for any subset of edges U, we denote by $V(U)$ the set $\cup_{e \in U} e$ of all endpoints of edges in U.

Lemma 75. *Let R be an optimal interdiction set and let M^* be a maximum cardinality matching in the graph $G[V \setminus R]$. Then the set $R^* \subseteq V$ defined below is also an optimal solution to the interdiction problem.*

$$R^* = V_\ell \setminus V(M^*),$$

where $\ell \in \{0, \ldots, n\}$ is the largest value such that $c(V_\ell \setminus V(M^*)) \leq B$.

Proof. The interdiction set R^* is budget feasible by assumption, and $R, R^* \subseteq V \setminus V(M^*)$ so $|R| \leq |R^*|$ by the construction of R^*. Let M' be a maximum cardinality matching in the graph $G[V \setminus R^*]$. Since M^* is a matching in $G[V \setminus R^*]$ it holds that $|M'| \geq |M^*|$. Thus R^* is also an optimal interdiction set because

$$\alpha(G[V \setminus R^*]) = |V| - |R^*| - |M'| \leq |V| - |R| - |M^*| = \alpha(G[V \setminus R]).$$

□

One of our key observations is that we can find an optimal matching M^* of Lemma 75 efficiently by matroid intersection techniques if we know the following four quantities that depend on M^*:

1. The maximum value $\ell \in \{0, \ldots, n\}$ such that $R^* = V_\ell \setminus V(M^*)$ satisfies $c(R^*) \leq B$;
2. $\beta_I = |V_\ell \cap V(M^*) \cap I|$;
3. $\beta_J = |V_\ell \cap V(M^*) \cap J|$;
4. $\gamma = |M^*|$.

There may be different quadruples $(\ell, \beta_I, \beta_J, \gamma)$ that correspond to different optimal matchings M^*. However, we need any such set of values that corresponds to an optimal M^*. Before showing how an optimal quadruple $(\ell, \beta_I, \beta_J, \gamma)$ can be used to find M^* by matroid intersection, we highlight that there is only a polynomial number of possible quadruples. This follows since $\ell \in \{0, \ldots, n\}$ can only take $n + 1$ different values, β_I and β_J only take at most $|I| + 1$ and $|J| + 1$ different values, respectively, and the cardinality of M^* is between 0 and $\nu(G) \leq \min\{|I|, |J|\}$. Hence, each possible quadruple $(\ell, \beta_I, \beta_J, \gamma)$ is element of the set

$$\mathcal{Q} = \{0, \ldots, n\} \times \{0, \ldots, |I| + 1\} \times \{0, \ldots, |J| + 1\} \times \{0, \ldots, \nu(G)\}.$$

We will go through all quadruples in $\mathcal{Q}$ and try to construct a corresponding mathcing M^* by the

matroid intersection technique that we introduce below. Thus, we will consider at least once an optimal quadruple, for which we will obtain an optimal M^*, which will then lead to an optimal R^* through Lemma 75. Hence, our task reduces to find a matching that "corresponds" to a given quadruple in $\mathcal{Q}$. We define formally what this means in the following.

Definition 76. We say that a matching M in G *corresponds* to $(\ell, \beta_I, \beta_J, \gamma) \in \mathcal{Q}$ if the following conditions are fulfilled:

1. $c(V_\ell \setminus V(M)) \leq B$,
2. $\beta_I = |V_\ell \cap V(M) \cap I|$,
3. $\beta_J = |V_\ell \cap V(M) \cap J|$,
4. $\gamma = |M|$.

We call a quadruple in $\mathcal{Q}$ *feasible* if there exists a matching that corresponds to it. Furthermore, a quadruple is called *optimal* if there is a matching M^* corresponding to it such that $R^* = V_\ell \setminus V(M^*)$ is an optimal interdiction set.

Notice that our definition of a matching M corresponding to a quadruple $(\ell, \beta_I, \beta_J, \gamma) \in \mathcal{Q}$ does not require that ℓ is the maximum value such that $c(V_\ell \setminus M) \leq B$ since we obtain the properties we need without requiring this condition in our correspondence, as shown by the next lemma.

Lemma 77. *Let $(\ell, \beta_I, \beta_J, \gamma) \in \mathcal{Q}$ be a feasible quadruple with M corresponding to it. Then the set $R = V_\ell \setminus V(M)$ is an interdiction set of objective value*

$$\alpha(G[V \setminus R]) \leq |V| - |R| - |M| = |V| - \gamma - \ell + \beta_I + \beta_J. \tag{4.22}$$

Furthermore, if $(\ell, \beta_I, \beta_J, \gamma) \in \mathcal{Q}$ is an optimal quadruple, then

$$\alpha(G[V \setminus R]) = |V| - \gamma - \ell + \beta_I + \beta_J.$$

Proof. The inequality in (4.22) follows immediately from (4.21) since

$$\begin{aligned} \alpha(G[V \setminus R]) &= |V| - |R| - \nu(G[V \setminus R]) && \text{(by (4.21))} \\ &\leq |V| - |R| - |M| && \text{(since } M \text{ is a matching in } G[V \setminus R]), \end{aligned}$$

with equality if and only if M is a maximum cardinality matching in $G[V \setminus R]$. To obtain the equality in (4.22), we observe that $|M| = \gamma$. Furthermore,

$$|R| = |V_\ell \setminus V(M)| = |V_\ell| - |V_\ell \cap V(M) \cap I| - |V_\ell \cap V(M) \cap J| = \ell - \beta_I - \beta_J,$$

which implies the desired equality. □

The main consequence of Lemma 77 is that the value of an optimal solution is determined entirely by its quadruple. Thus one can find an optimal quadruple by testing the feasibility of every quadruple in $\mathcal{Q}$ and choosing one that minimizes the right hand side of (4.22). We conclude the following.

Corollary 78. *Let $(\ell, \beta_I, \beta_J, \gamma) \in \mathcal{Q}$ be a feasible quadruple that minimizes $|V| - \gamma - \ell + \beta_I + \beta_J$, and let M^* be a matching that corresponds to it. Then $R^* = V_\ell \setminus V(M^*)$ is an optimal interdiction set to bipartite stable set interdiction problem.*

Hence, all that remains to be done to obtain an efficient algorithm is to design a procedure that, for a quadruple $(\ell, \beta_I, \beta_J, \gamma) \in \mathcal{Q}$ decides whether it is feasible, and if so, finds a corresponding matching M. Using this procedure we check all quadruples to determine a feasible quadruple $(\ell, \beta_I, \beta_J, \gamma)$ that minimizes $|V| - \gamma - \ell - \beta_I - \beta_J$, and then return $R^* = V_\ell \setminus V(M^*)$, where M^* is a matching corresponding to such a quadruple.

Hence, let $q = (\ell, \beta_I, \beta_J, \gamma) \in Q$ and we show how to check feasibility of q and find a corresponding matching M if q is feasible. Let $c' : E \to \mathbb{Z}_{\geq 0}$ be an auxiliary cost function defined

by

$$c'(v_k) = \begin{cases} c(v_k) & \text{if } k \leq \ell, \\ 0 & \text{if } k > \ell. \end{cases}$$

Based on c' we define weights $w : E \to \mathbb{Z}_{\geq 0}$, where

$$w(\{i,j\}) = c'(i) + c'(j) \qquad \forall \{i,j\} \in E.$$

Our goal is to determine a maximum weight matching M in G such that $|V_\ell \cap V(M) \cap I| = \beta_I$, $|V_\ell \cap V(M) \cap J| = \beta_J$, and $\gamma = |M|$. Notice that maximizing w corresponds to maximizing $c(V(M) \cap V_\ell)$. Hence, a maximizer M will be a matching in G that satisfies conditions (2)-(4) of Definition 76 and subject to fulfilling these three conditions, it maximizes $c(V(M) \cap V_\ell)$, which is the same as minimizing $c(V_\ell \setminus V(M))$. Hence, if $c(V_\ell \setminus V(M)) \leq B$, then the quadruple $(\ell, \beta_I, \beta_J, \gamma)$ is feasible and M corresponds to it, otherwise, the quadruple is not feasible. It remains to show how to find efficiently a maximum weight matching M in G such that $|V_\ell \cap V(M) \cap I| = \beta_I$, $|V_\ell \cap V(M) \cap J| = \beta_J$, and $\gamma = |M|$.

This optimization problem corresponds to maximizing w over a face of a matroid intersection polytope. Indeed, define one laminar matroid $M_1 = (E, \mathcal{F}_1)$ such that a set $U \subseteq E$ is independent in M_1, i.e., $U \in \mathcal{F}_1$, if U contains at most one edge incident with $i \in I$ for each $v \in I$, at most β_I edges incident with vertices in $V_\ell \cap I$ and at most γ edges in total. Similarly, define $M_2 = (E, \mathcal{F}_2)$ such that $U \in \mathcal{I}_2$ if U contains am most one edge incident with any vertex $j \in J$, at most β_J edges incident with $V_\ell \cap J$ and at most γ edges in total. The problem we want to solve is to find a set $M \in \mathcal{F}_1 \cap \mathcal{F}_2$ such that the constraints $|M \cap V_\ell \cap I| \leq \beta_I$, $|M \cap V_\ell \cap J| \leq \beta_J$ and $|M| \leq \gamma$ are fulfilled with equality. Hence, this is indeed the problem of maximizing w over a particular face of the matroid intersection polytope corresponding to M_1 and M_2. This problem can be solved in strongly polynomial time by matroid intersection algorithms. Alternatively, one can also find a vertex solution to the following polynolmial-sized LP, which describes this face of the

matroid intersection polytope, and is therefore integral when feasible.

$$\begin{aligned} \max \quad & w^T x & & \\ & x(\delta(v)) & \leq \quad 1 & \quad \forall v \in V \\ & x(\delta(V_\ell \cap I)) & = \quad \beta_I & \\ & x(\delta(V_\ell \cap J)) & = \quad \beta_J & \\ & x(E) & = \quad \gamma & \end{aligned}$$

For more details on optimization over the matroid intersection polytope, we refer the interested reader to [93, Chapter 41].

4.6 Conclusions

We presented a framework to obtain 2-pseudoapproximations for a wide set of combinatorial interdiction problems, including maximum cardinality independent set in a matroid or the intersection of two matroids, maximum s-t flows, and packing problems defined by a constraint matrix that is TU. Our approach is inspired by a technique of Burch et al. [26], who presented a 2-pseudoapproximation for maximum s-t flows. Furthermore, we show that our framework can also be adapted to more general settings involving matroid optimization. More precisely, we also get a 2-pseudpapproximation for interdicting the maximum weight independent set problem in a matroid with submodular interdiction costs. Submodularity is a natural property for interdiction costs since it models economies of scale. Our framework for 2-pseudoapproximations is polyhedral and sometimes we can exploit polyhedral properties of well-structured interdiction problems to obtain stronger results. We demonstrate this on the problem of interdicting b-stable sets in bipartite graphs. For this setting we obtain a PTAS, by employing ideas from multi-budgeted optimization. Furthermore, we show that the special case of stable set interdiction in bipartite graphs can be solved efficiently by matroid intersection techniques.

Many interesting open questions remain in the field of interdicting combinatorial optimization problems. It particular, it remains open whether stronger pseudoapproximations can be obtained for the considered problems. Also in terms of "true" approximation algorithms, relatively little is known. Two of the most prominent open problems are whether there are constant-factor approximations for interdicting maximum s-t flows and minimum spanning trees.

Bibliography

[1] Ryan Admiraal and Mark S Handcock. Networksis: a package to simulate bipartite graphs with fixed marginals through sequential importance sampling. *Journal of Statistical Software*, 24(8):1–21, 2008.

[2] R. K. Ahuja, T.L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1993.

[3] Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. In *Proceedings of the 28th Annual ACM Symposium on the Theory of Computing*, pages 20–29. ACM, New York, 1996.

[4] Alexandr Andoni, Robert Krauthgamer, and Krzysztof Onak. Streaming algorithms via precision sampling. In *Foundations of Computer Science (FOCS), 2011 IEEE 52nd Annual Symposium on*, pages 363–372. IEEE, 2011.

[5] Alexandr Andoni, Huy L. Nguyễn, Yury Polyanskiy, and Yihong Wu. Tight lower bound for linear sketches of moments. In *Proceedingns of the 40th International Colloquium on Automata, Languages and Programming*. Springer, 2013.

[6] N. Assimakopoulos. A network interdiction model for hospital infection control. *Computers in biology and medicine*, 17(6):413–422, 1987.

[7] ATLAS Experiment. What is ATLAS? `http://atlas.ch/what_is_atlas.html`, 2014. [Online; accessed 20-December-2014].

[8] Brian Babcock, Shivnath Babu, Mayur Datar, Rajeev Motwani, and Jennifer Widom. Models and issues in data stream systems. In *Proceedings of the 21st ACM Symposium on Principles of Database Systems*, 2002.

[9] M. O. Ball, B. Golden, and R. V. Vohra. Finding the most vital arcs in a network. *Operations Research Letters*, 8(2):73–76, 1989.

[10] Ziv Bar-Yossef. *The complexity of massive data set computations*. PhD thesis, University of California, 2002.

[11] Ziv Bar-Yossef, T. S. Jayram, Ravi Kumar, and D. Sivakumar. An information statistics approach to data stream and communication complexity. *J. Comput. System Sci.*, 68(4):702–732, 2004.

[12] Alexander Barvinok. On the number of matrices and a random matrix with prescribed row and column sums and 0–1 entries. *Advances in Mathematics*, 224(1):316–339, 2010.

[13] C. Bazgan, S. Toubaline, and Z. Tuza. The most vital nodes with respect to independent set and vertex cover. *Discrete Applied Mathematics*, 159(17):1933 – 1946, 2011.

[14] C. Bazgan, S. Toubaline, and D. Vanderpooten. Complexity of determining the most vital elements for the p-median and p-center location problems. *Journal of Combinatorial Optimization*, 25(2):191–207, 2013.

[15] A. Berger, V. Bonifaci, F. Grandoni, and G. Schäfer. Budgeted matching and budgeted matroid intersection via the gasoline puzzle. In *Integer Programming and Combinatorial Optimization*, pages 273–287. Springer, 2008.

[16] Ivona Bezáková, Nayantara Bhatnagar, and Dana Randall. On the diaconis-gangolli markov chain for sampling contingency tables with cell-bounded entries. *Journal of combinatorial optimization*, 22(3):457–468, 2011.

[17] Ivona Bezáková, Nayantara Bhatnagar, and Eric Vigoda. Sampling binary contingency tables with a greedy start. *Random Structures and Algorithms*, 30(1-2):168–205, 2007.

[18] Ivona Bezáková, Alistair Sinclair, Daniel Štefankovič, and Eric Vigoda. Negative examples for sequential importance sampling of binary contingency tables. In *Proceedings of the European Symposium on Algorithms*, pages 136–147. Springer, 2006.

[19] Lakshminath Bhuvanagiri, Sumit Ganguly, Deepanjan Kesh, and Chandan Saha. Simpler algorithm for estimating frequency moments of data streams. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 708–713. ACM, 2006.

[20] Jose Blanchet and Alexandre Stauffer. Characterizing optimal sampling of binary contingency tables via the configuration model. *Random Structures & Algorithms*, 42(2):159–184, 2013.

[21] Béla Bollobás. A probabilistic proof of an asymptotic formula for the number of labelled regular graphs. *European J. Combin.*, 1(4):311–316, 1980.

[22] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004.

[23] Vladimir Braverman, Jonathan Katzman, Charles Seidell, and Gregory Vorsanger. An optimal algorithm for large frequency moments using $O(n^{1-2/k})$ bits. *Approximation, Randomization, and Combinatorial Optimization*, 28:531–544, 2014.

[24] Vladimir Braverman and Rafail Ostrovsky. Recursive sketching for frequency moments. *arXiv preprint arXiv:1011.2571*, 2010.

[25] Vladimir Braverman and Rafail Ostrovsky. Zero-one frequency laws. In *Proceedings of the 42nd Annual ACM Symposium on the Theory of Computing*, pages 281–290. ACM, New York, 2010.

[26] C. Burch, R. Carr, S. Krumke, M. Marathe, C. Phillips, and E. Sundberg. A decomposition-based pseudoapproximation algorithm for network flow inhibition. In *Network Interdiction and Stochastic Integer Programming*, pages 51–68. Springer, 2003.

[27] CERN. LHC Brochure. `http://cds.cern.ch/record/1092437/files/CERN-Brochure-2008-001-Eng.pdf`, (accessed January 29, 2014).

[28] Amit Chakrabarti, Graham Cormode, and Andrew McGregor. A near-optimal algorithm for computing the entropy of a stream. In *Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 328–335. ACM, New York, 2007.

[29] Amit Chakrabarti, Khanh Do Ba, and S. Muthukrishnan. Estimating entropy and entropy norm on data streams. *Internet Math.*, 3(1):63–78, 2006.

[30] Amit Chakrabarti, Subhash Khot, and Xiaodong Sun. Near-optimal lower bounds on the multi-party communication complexity of set disjointness. In *Proceedings of the 18th Annual IEEE Conference on Computational Complexity*, pages 107–117, 2003.

[31] Amit Chakrabarti and Oded Regev. An optimal lower bound on the communication complexity of gap-Hamming-distance. *SIAM J. Comput.*, 41(5):1299–1317, 2012.

[32] Moses Charikar, Kevin Chen, and Martin Farach-Colton. Finding frequent items in data streams. In *Automata, languages and programming*, volume 2380 of *Lecture Notes in Comput. Sci.*, pages 693–703. Springer, Berlin, 2002.

[33] Yuguo Chen, Persi Diaconis, Susan P Holmes, and Jun S Liu. Sequential Monte Carlo methods for statistical analysis of tables. *Journal of the American Statistical Association*, 100(469):109–120, 2005.

[34] F. R. K. Chung, R. L. Graham, and S.-T. Yau. On sampling with Markov chains. *Random Structures Algorithms*, 9(1-2):55–77, 1996.

[35] R. L. Church, M. P. Scaparra, and R. S. Middleton. Identifying critical infrastructure: the median and covering facility interdiction problems. *Annals of the Association of American Geographers*, 94(3):491–502, 2004.

[36] M. Conforti, G. Cornuéjols, and G. Zambelli. Extended formulations in combinatorial optimization. *Annals of Operations Research*, 204(1):97–143, 2013.

[37] Don Coppersmith and Ravi Kumar. An improved data stream algorithm for frequency moments. In *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '04, pages 151–156, Philadelphia, PA, USA, 2004. Society for Industrial and Applied Mathematics.

[38] Graham Cormode and S. Muthukrishnan. An improved data stream summary: the count-min sketch and its applications. *J. Algorithms*, 55(1):58–75, 2005.

[39] Persi Diaconis and Anil Gangolli. *Rectangular arrays with fixed margins*, volume 72 of *IMA Vol. Math. Appl.* Springer, New York, 1995.

[40] Persi Diaconis and Bernd Sturmfels. Algebraic algorithms for sampling from conditional distributions. *Ann. Statist.*, 26(1):363–397, 1998.

[41] M. Dinitz and A. Gupta. Packing interdiction and partial covering problems. In *Integer Programming and Combinatorial Optimization*, pages 157–168. Springer, 2013.

[42] Martin Dyer, Ravi Kannan, and John Mount. Sampling contingency tables. *Random Structures Algorithms*, 10(4):487–506, 1997.

[43] G. N. Frederickson and R. Solis-Oba. Increasing the weight of minimum spanning trees. In *Proceedings of the seventh annual ACM-SIAM symposium on Discrete algorithms*, pages 539–546. Society for Industrial and Applied Mathematics, 1996.

[44] G. N. Frederickson and R. Solis-Oba. Efficient algorithms for robustness in matroid optimization. In *Proceedings of the eighth annual ACM-SIAM symposium on Discrete algorithms*, pages 659–668. Society for Industrial and Applied Mathematics, 1997.

[45] David Gale. A theorem on flows in networks. *Pacific J. Math.*, 7:1073–1082, 1957.

[46] Sumit Ganguly. Estimating frequency moments of data streams using random linear combinations. In *Approximation, Randomization, and Combinatorial Optimization*, pages 369–380. Springer, 2004.

[47] Sumit Ganguly. Polynomial estimators for high frequency moments. *arXiv preprint arXiv:1104.4552*, 2011.

[48] F. Grandoni, R. Ravi, M. Singh, and R. Zenklusen. New approaches to multi-objective optimization. *Mathematical Programming, Series A*, 146(1):525–554, 2014.

[49] André Gronemeier. Asymptotically optimal lower bounds on the NIH-multi-party information complexity of the AND-function and disjointness. In *Proceedings of the 26th Annual International Symposium on Theoretical Aspects of Computer Science*, 2009.

[50] Yunhong Gu. Google Public DNS now supports DNSSEC validation. `http://googleonlinesecurity.blogspot.com/2013/03/google-public-dns-now-supports-dnssec.html`, 2013 (accessed January 29, 2014).

[51] Sudipto Guha, Piotr Indyk, and Andrew McGregor. Sketching information divergences. In *Learning theory*, volume 4539 of *Lecture Notes in Comput. Sci.*, pages 424–438. Springer, Berlin, 2007.

[52] Matthew T Harrison and Jeffrey W Miller. Importance sampling for weighted binary random matrices with specified margins. *arXiv preprint arXiv:1301.3928*, 2013.

[53] Nicholas JA Harvey, Jelani Nelson, and Krzysztof Onak. Sketching and streaming entropy via approximation theory. In *Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science*, pages 489–498, 2008.

[54] C. A. J. Hurkens. On the diameter of the edge cover polytope. *Journal of Combinatorial Theory, Series B*, 51(2):271–276, 1991.

[55] Piotr Indyk. Algorithms for dynamic geometric problems over data streams. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing*, pages 373–380. ACM, New York, 2004.

[56] Piotr Indyk. Stable distributions, pseudorandom generators, embeddings, and data stream computation. *Journal of the ACM*, 53(3):307–323, 2006.

[57] Piotr Indyk, Andrew McGregor, Ilan Newman, and Krzysztof Onak. Open problems in data streams, property testing, and related topics, 2011. Bertinoro Workshop on Sublinear Algorithms (2011) and IITK Workshop on Algorithms for Processing Massive Data Sets (2009).

[58] Piotr Indyk and David Woodruff. Tight lower bounds for the distinct elements problem. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, pages 283–288. IEEE, 2003.

[59] Piotr Indyk and David Woodruff. Optimal approximations of the frequency moments of data streams. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing*, pages 202–208. ACM, New York, 2005.

[60] Mark Jerrum and Alistair Sinclair. Approximating the permanent. *SIAM J. Comput.*, 18(6):1149–1178, 1989.

[61] Mark Jerrum and Alistair Sinclair. Fast uniform generation of regular graphs. *Theoret. Comput. Sci.*, 73(1):91–100, 1990.

[62] Mark Jerrum and Alistair Sinclair. The Markov chain Monte Carlo method: an approach to approximate counting and integration. In *Approximation Algorithms for NP-hard problems*, pages 482–520. PWS Publishing Co., 1996.

[63] William B Johnson and Joram Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. *Contemporary mathematics*, 26(189-206):1, 1984.

[64] G. Joret and A. Vetta. Reducing the rank of a matroid. *arXiv preprint arXiv:1211.4853*, 2012.

[65] Hossein Jowhari, Mert Sağlam, and Gábor Tardos. Tight bounds for lp samplers, finding duplicates in streams, and related problems. In *Proceedings of the 13th ACM Symposium on Principles of Database Systems*, pages 49–58. ACM, 2011.

[66] A. Jüttner. On budgeted optimization problems. *SIAM Journal on Discrete Mathematics*, 20(4):880–892, 2006.

[67] V. Kaibel. Extended formulations in combinatorial optimization. *Optima*, 85:2–7, 2011.

[68] Daniel M. Kane, Jelani Nelson, and David P. Woodruff. On the exact space complexity of sketching and streaming small norms. In *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1161–1178. SIAM, Philadelphia, PA, 2010.

[69] Daniel M Kane, Jelani Nelson, and David P Woodruff. An optimal algorithm for the distinct elements problem. In *Proceedings of the 29th Annual ACM Symposium on Principles of Database Systems*, pages 41–52, 2010.

[70] Ravi Kannan, Prasad Tetali, and Santosh Vempala. Simple Markov-chain algorithms for generating bipartite graphs and tournaments (extended abstract). In *Proceedings of the 8th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 193–200. ACM, New York, 1997.

[71] L. Khachiyan, E. Boros, K. Borys, K. Elbassioni, V. Gurvich, G. Rudolf, and J. Zhao. On short paths interdiction problems: total and node-wise limited interdiction. *Theory of Computing Systems*, 43(2):204–233, 2008.

[72] Valerie King, Satish Rao, and Rorbert Tarjan. A faster deterministic maximum flow algorithm. *Journal of Algorithms*, 17(3):447–474, 1994.

[73] B. Korte and J. Vygen. *Combinatorial Optimization, Theory and Algorithms*. Springer, 5th edition, 2012.

[74] Raffi Krikorian. New Tweets per second record, and how! `https://blog.twitter.com/2013/new-tweets-per-second-record-and-how`, 2013 (accessed January 29, 2014).

[75] Eyal Kushilevitz and Noam Nisan. *Communication complexity*. Cambridge University Press, Cambridge, 1997.

[76] Anukool Lakhina, Mark Crovella, and Christophe Diot. Mining anomalies using traffic feature distributions. In *ACM SIGCOMM Computer Communication Review*, volume 35, pages 217–228. ACM, 2005.

[77] Yi Li and David P. Woodruff. A tight lower bound for high frequency moment estimation with small error. In *Approximation, Randomization, and Combinatorial Optimization*, volume 8096 of *Lecture Notes in Comput. Sci.*, pages 623–638. Springer, Heidelberg, 2013.

[78] Bryan FJ Manly. A note on the analysis of species co-occurrences. *Ecology*, 76:1109–1115, 1995.

[79] Albert W Marshall, Ingram Olkin, and Barry C Arnold. *Inequalities: Theory of Majorization and Its Applications: Theory of Majorization and Its Applications*. Springer, 2010.

[80] Andrew McGregor, Atri Rudra, and Steve Uurtamo. Polynomial fitting of data streams with applications to codeword testing. In *Proceedingns of the 28th International Symposium on*

Theoretical Aspects of Computer Science. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2011.

[81] N. Megiddo. Combinatorial optimization with rational objective functions. *Mathematics of Operations Research*, 4(4):414–424, November 1979.

[82] Jeffrey W. Miller and Matthew T. Harrison. Exact sampling and counting for fixed-margin matrices. *The Annals of Statistics*, 41(3):1569–1592, 06 2013.

[83] Morteza Monemizadeh and David P Woodruff. 1-pass relative-error l_p-sampling with applications. In *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1143–1160. Society for Industrial and Applied Mathematics, 2010.

[84] S Muthukrishnan. *Data streams: Algorithms and applications*. Now Publishers Inc, 2005.

[85] James B Orlin. Max flows in $o(nm)$ time, or better. In *Proceedings of the 45th Annual ACM Symposium on Theory of Computing*, pages 765–774. ACM, 2013.

[86] F. Pan, W. S. Charlton, and D. P. Morton. A stochastic program for interdicting smuggled nuclear material. In *Network interdiction and stochastic integer programming*, pages 1–19. Springer, 2003.

[87] F. Pan and A. Schild. Interdiction problems on planar graphs. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 317–331. Springer, 2013.

[88] C. A. Phillips. The network inhibition problem. In *Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, pages 776–785. ACM, 1993.

[89] R. Ravi and M. X. Goemans. The constrained minimum spanning tree problem. In *Proceedings of 5th Scandinavian Workshop on Algorithm Theory (SWAT)*, pages 66–75, 1996.

[90] H. J. Ryser. Combinatorial properties of matrices of zeros and ones. *Canad. J. Math.*, 9:371–377, 1957.

[91] J. Salmeron, K. Wood, and R. Baldick. Worst-case interdiction analysis of large-scale electric power grids. *Power Systems, IEEE Transactions on*, 24(1):96–104, 2009.

[92] A. Schrijver. On the history of the transportation and maximum flow problems. *Mathematical Programming*, 91(3):437–445, 2002.

[93] A. Schrijver. *Combinatorial optimization: polyhedra and efficiency*, volume 24. Springer, 2003.

[94] L. G. Valiant. The complexity of computing the permanent. *Theoret. Comput. Sci.*, 8(2):189–201, 1979.

[95] R. D. Wollmer. Removing arcs from a network. *Operations Research*, 12(6):934–940, 1964.

[96] R. K. Wood. Deterministic network interdiction. *Mathematical and Computer Modelling*, 17(2):1–18, 1993.

[97] David Woodruff. Optimal space lower bounds for all frequency moments. In *Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 167–175, 2004.

[98] Nicholas C. Wormald. Generating random regular graphs. *J. Algorithms*, 5(2):247–280, 1984.

[99] R. Zenklusen. Matching interdiction. *Discrete Applied Mathematics*, 158(15):1676–1690, 2010.

[100] R. Zenklusen. Network flow interdiction on planar graphs. *Discrete Applied Mathematics*, 158(13):1441–1455, 2010.

[101] R. Zenklusen. Connectivity interdiction. *Operations Research Letters*, 42(67):450 – 454, 2014.

Vita

Stephen Robert Chestnut attended Southwest Guilford High School in High Point, North Carolina, and, afterwards, Cornell University of Ithaca, New York. He graduated from Cornell in 2004 with a Bachelor of Science degree in Engineering Physics. Beginning in November of that year, he worked in the title of Scientist at ENSCO, Inc., a mid-sized science and technology development firm in Washington, D.C., metropolitan area. At ENSCO he was primarily developing algorithms and software for navigation equipment to be used by first responders and in other specialized applications. While there he volunteered with Team ENSCO, a company sponsored entrant into the 2005 DARPA Grand Challenge autonomous vehicle race.

Stephen left industry in August of 2008 to enroll in the Master of Science program in the Department of Applied Mathematics at the University of Colorado. In the summer of 2009, he participated in the National Science Foundation EAPSI program, spending the summer in Taipei, Taiwan, at Academia Sinica, and in 2010 he was awarded the Maurice Davies Award by the the Colorado/Wyoming Chapter of the American Statistical Association. He completed his Master's degree in 2010, and

enrolled in the Ph.D. program of the Applied Mathematics and Statistics Department at Johns Hopkins University where he was supported as a GAANN Fellow during 2010-2013.

At Johns Hopkins, Stephen has taught Introduction to Statistics, Discrete Mathematics, and Probability and Statistics and has also been a teaching assistant for eleven other courses. He received a Centennial Award in 2013, has thrice been awarded the Professor Joel Dean Award for Excellence in Teaching, and has thrice been awarded a Campbell Fellowship. His research interests are in combinatorial optimization, theoretical computer science, and discrete probability. In April 2015, Stephen will be joining the Institute for Operations Research at ETH Zürich in Zürich, Switzerland, as a postdoctoral researcher.