Probabilistic Methods in Combinatorial and Stochastic Optimization

by

Jan Vondrák

Submitted to the Department of Mathematics
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2005

Author .....
Department of Mathematics
January 14, 2005

Certified by .....
Michel X. Goemans
Professor of Applied Mathematics
Thesis Supervisor

Accepted by .....
Rodolfo Ruben Rosales
Chairman, Applied Mathematics Committee

Probabilistic Methods in Combinatorial and Stochastic Optimization

by

Jan Vondrák

Submitted to the Department of Mathematics
on January 14, 2005, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

In this thesis we study a variety of combinatorial problems with inherent randomness. In the first part of the thesis, we study the possibility of covering the solutions of an optimization problem on random subgraphs. The motivation for this approach is a situation where an optimization problem needs to be solved repeatedly for random instances. Then we seek a pre-processing stage which would speed-up subsequent queries by finding a fixed sparse subgraph covering the solution for a random subgraph with high probability. The first problem that we investigate is the minimum spanning tree. Our essential result regarding this problem is that for every graph with edge weights, there is a set of $O(n \log n)$ edges which contains the minimum spanning tree of a random subgraph with high probability. More generally, we extend this result to matroids. Further, we consider optimization problems based on the shortest path metric and we find covering sets of size $O(n^{1+2/c} \log^2 n)$ that approximate the shortest path metric of a random vertex-induced subgraph within a constant factor of c with high probability.

In the second part, we turn to a model of stochastic optimization, where a solution is built sequentially by selecting a collection of "items". We distinguish between adaptive and non-adaptive strategies, where adaptivity means being able to perceive the precise characteristics of chosen items and use this knowledge in subsequent decisions. The benefit of adaptivity is our central concept which we investigate for a variety of specific problems. For the Stochastic Knapsack problem, we prove constant upper and lower bounds on the "adaptivity gap" between optimal adaptive and non-adaptive policies. For more general Stochastic Packing/Covering problems, we prove upper and lower bounds on the adaptivity gap depending on the dimension. We also design polynomial-time algorithms achieving near-optimal approximation guarantees with respect to the adaptive optimum. Finally, we prove complexity-theoretic results regarding optimal adaptive policies. These results are based on a connection between adaptive policies and Arthur-Merlin games which yields PSPACE-hardness results for numerous questions regarding adaptive policies.

Thesis Supervisor: Michel X. Goemans
Title: Professor of Applied Mathematics

Acknowledgments

First and foremost, my sincere thanks go to my advisor, Michel Goemans. His influence permeates every part of this thesis, from informal discussions, invaluable suggestions and ideas, through brain-storming sessions in his office, to a never-ending challenge to push the limits of my ability. My thanks go to Michel not only for being my academic advisor but also for being a wise mentor and cheerful companion throughout my years at MIT, never confining his support to the area of academics. Thanks for all the time you gave me, knowing how scarce your time can be!

I would also like to thank Santosh Vempala, for his inspiring classes, many helpful discussions and motivating questions (which inspired especially Chapter 4 of this thesis). My thanks go to many other professors at MIT, that I had the opportunity to work with or learn from. I would like to name especially Dan Kleitman, Igor Pak, Rom Pinchasi, Peter Shor, Mike Sipser, Dan Spielman, David Karger, Piotr Indyk, Arthur Mattuck and Hartley Rogers. Big thanks to János Pach at NYU, for recent collaboration and generous support. Thanks to other collaborators, namely Nicole Immorlica, without whom the first half of my thesis would never get started, and Brian Dean, without whom the second half of my thesis would never get started, Radoš Radoičić, whose merits go beyond being a collaborator, and finally Tim Chow and Kenneth Fan.

I should not forget to express my gratitude to those back home who made it possible for me to come to MIT and who taught me many things that still help my brain tick. Foremost, my advisor in Prague, Martin Loebl, for all his support, unceasing mathematical curiosity and indomitable optimism. A lot of thanks to Jirka Matoušek for his wonderful lectures and discussions. Thanks to both of you for your barely warranted belief in me. Thanks to Jaroslav Nešetřil, Jan Kratochvíl and many others for creating an amazing academic environment at KAM MFF UK. And my deepest gratitude to my parents in Prague, my brother Šipak, and the rest of the family, for being always there for me.

During my years at MIT, I met many friends who made my years in Boston great fun and a wonderful experience. Thanks to Sergi for being my friend and colleague in math from Day 1 in Boston. Thanks to Radoš for his uplifting sarcasm, unyielding criticism and profound advice in many life situations. Thanks to Fumei for her smiling support in all these years. Thanks to all my friends in the department: Brad, Brian, Ian, Kevin, Jaheyuk, Boguk and Vahab. Best regards to old man Young Han. Many thanks to others who helped create the great atmosphere at MIT and who are now scattered around the world: José, Mohammad, Kari, Wai Ling and others. Outside of MIT, I would like to thank all the people who were my companions at times cheerful or difficult: Rad, Yurika, David, Summer and Sun; Pavel, Alois, Jirka and Ying; Hazhir and Anya; Kalina and Samantha. Thanks to Martin for being a big friend, roommate and dangerous influence in the last two years.

And finally, thank you, Maryam, for making my last year of graduate school the happiest one.

Contents

Chapter 1

Introduction

In this thesis we study a variety of combinatorial problems. The common feature of these problems is their *stochastic aspect*, i.e. inherent randomness affecting the input. We investigate two essentially different scenarios.

In the first setting, we study the possibility of *covering the solutions* of an optimization problem on random subgraphs. The motivation for this approach is a situation where an optimization problem needs to be solved repeatedly for random instances. Imagine a company that needs to solve an optimization problem, such as the Vehicle Routing Problem, every day. The requests vary randomly but the underlying structure of the problem remains the same. In this kind of situation, we seek a pre-processing stage which would speed-up subsequent queries by finding a fixed sparse subgraph covering the solution for a random subgraph with high probability.

In the second setting, we study stochastic optimization problems with uncertainty on the input, where a solution is built sequentially by selecting a collection of "items". An example of such an optimization problem is machine scheduling where jobs should be assigned to machines but it is not a priori known how long each job takes to complete. However, once a job is completed, we know exactly its running time. We distinguish between *adaptive* and *non-adaptive* strategies, where adaptivity means being able to perceive the precise properties of chosen items (in this case, the running time of completed jobs), and use this knowledge in subsequent decisions. The benefit of adaptivity is our central concept which we investigate for a variety of specific problems. We also study the question of algorithmic approximation of the adaptive optimum and related complexity-theoretic issues.

In the following, we summarize the most important results of this thesis.

1.1 Optimization problems on random subgraphs

The first half of this thesis concerns solutions of combinatorial problems on random subgraphs. With the goal of solving a given problem repeatedly for random subgraphs, we would like to find a sparse subgraph which contains the solution with high probability. The first problem we consider is a prototypical graph optimization problem, the Minimum Spanning Tree.

Covering minimum spanning trees of random subgraphs. A conjecture proposed by Michel Goemans in 1990 states that in any weighted complete graph, there is a subset Q of $O(n \log n)$ edges such that the minimum spanning tree of a random vertex-induced subgraph is contained in Q with high probability. We prove this conjecture and additional related results. The bound of $O(n \log n)$ is asymptotically optimal, even if we want to cover the minimum spanning tree only with a constant probability.

Note that for a general graph G, a vertex-induced subgraph need not be connected. However, this does not change anything in our analysis. In case an induced subgraph $G[W]$ is not connected, we consider the minimum spanning forest, i.e. the minimum spanning tree on each component. We prove the following.

Theorem 1. *Let G be a weighted graph on n vertices, $0 < p < 1$, $b = 1/(1-p)$ and $c > 0$. Let $V(p)$ denote a random subset of V where each vertex is present independently with probability p. Let $MST(W)$ denote the minimum spanning forest of the subgraph $G[W]$ induced by W. Then there exists a set $Q \subseteq E$ of size*

$$|Q| \leq e(c+1)n \log_b n + O(n)$$

such that for a random $W = V(p)$, $\Pr[MST(W) \subseteq Q] > 1 - \frac{1}{n^c}$.

This can be also viewed as a *network reliability* result: in any network, there are $O(n \log n)$ edges which suffice to ensure that the network remains connected with high probability, when some vertices fail randomly. Moreover, the connection is achieved by the minimum spanning tree in the remaining subgraph.

Minimum-weight bases in matroids. We prove a similar statement in the case of random edge-induced subgraphs. This result generalizes to any weighted matroid. In the following, $X(p)$ denotes a random subset of X where every element is sampled independently with probability p.

Theorem 2. *For any matroid of rank n on ground set X, with weighted elements, there exists a set $Q \subseteq X$ of size $O(n \log_b n)$, where $b = 1/(1-p)$, such that the minimum-weight basis of $X(p)$ is contained in Q with high probability.*

Note that the size of the covering set depends only on the rank of the matroid, and not on the size of the ground set.

Adversarial vertex removal. In contrast to random subgraphs, we can consider the model of a malicious adversary who chooses a subgraph knowing our covering set Q. Using our probabilistic methods, we can actually find a set Q which contains $MST(W)$ for *all subsets* W containing at least $n - k$ vertices.

Theorem 3. *For any graph G on n vertices and a fixed $k > 0$, there is a set of edges Q of size $|Q| < e(k+1)n$ such that for any $W \subseteq V$, $|W| \geq n - k$, we have $MST(W) \subseteq Q$.*

There are examples of weighted graphs where such a set Q must contain at least $n + (n-1) + \ldots + (n-k)$ vertices. We conjecture this to be the best upper bound but the probabilistic method does not seem able to yield such a tight bound. It is an interesting open question whether this or a similar statement has a purely combinatorial proof.

A priori optimization. The covering sets in all cases can be found efficiently by a randomized algorithm. Therefore, such covering sets can be used to speed up an algorithm solving the MST problem, in case the instances are drawn randomly from a large fixed graph. A single preprocessing step would yield a sparse subgraph of significantly reduced size which would allow more efficient responses to subsequent queries.

Metric approximation. Another result along the lines of a priori optimization concerns combinatorial problems based on the *shortest path metric* in a graph. Examples of such problems are the *s-t* Shortest Path, Steiner tree, Facility Location or Traveling Salesman. With the goal of solving these problems repeatedly for random subgraphs, we would like to have a sparse set of edges which preserves the shortest-path metric for a random subgraph with high probability. This question was posed by Santosh Vempala.

Here it is impossible to preserve the metric *exactly* with a significantly reduced number of edges. But we can preserve the metric *approximately*. This is related to the notion of a *spanner* - a graph approximately preserving the metric of a given graph. However, our conditions are stronger: we would like to have a set of edges which approximates all distances within a constant factor c with high probability, when vertices or edges fail at random. We call such a set of edges *c-metric-approximating*. We establish the following:

Theorem 4. *For any graph G with edge lengths, there is a c-metric-approximating set $Q \subseteq E$ (for random subgraphs induced by $W = V(p)$) such that*

$$|Q| = O\left(p^{-2c} n^{1+2/c} \log^2 n\right).$$

Theorem 5. *For any graph G with edge lengths, there is a c-metric-approximating set $Q \subseteq E$ (for random subgraphs induced by $F = E(p)$) such that*

$$|Q| = O\left(p^{-c} n^{1+2/c} \log n\right).$$

The proofs are constructive and the covering sets can be found efficiently. Similarly to lower bounds on the size of spanners, there is a lower bound of $\Omega(n^{1+1/c})$ on the size of Q, based on extremal graphs of girth $c+2$. There is a gap between this lower bound and the estimate on $|Q|$ but this is largely due to the lack of knowledge concerning extremal graphs of given girth. If, for example, the extremal number of edges for girth $c+2$ were to be improved to $O(n^{1+1/c})$, our bound in Theorem 4 would improve accordingly to $|Q| = O(n^{1+1/c} \log^2 n)$ (for constant p). Essentially, we pay the price of a logarithmic factor for metric approximation on random subgraphs.

Summary. Let's review the results for covering solutions of minimum spanning tree and metric-based problems for random subgraphs. The table shows the asymptotic bounds for random subgraphs induced by a random subset of vertices $V(p)$ or edges $E(p)$; here, we assume a constant sampling probability $p \in (0, 1)$. The details for MST covering can be found in Chapter 3. Metric approximation for random subgraphs is discussed in Chapter 4.

PROBLEM VARIANT	lower bound vertex-case	upper bound vertex-case	lower bound edge-case	upper bound edge-case
metric $c \in [1, 3)$	$\Omega\left(n^2\right)$	$O\left(n^2\right)$	$\Omega\left(n^2\right)$	$O\left(n^2\right)$
metric $c \in [3, 5)$	$\Omega\left(n^{3/2}\right)$	$O\left(n^{3/2} \log n\right)$	$\Omega\left(n^{3/2}\right)$	$O\left(n^{3/2} \log n\right)$
metric $c \geq 5$	$\Omega\left(n^{1+1/c}\right)$	$O\left(n^{1+2/c} \log^2 n\right)$	$\Omega\left(n^{1+1/c}\right)$	$O\left(n^{1+2/c} \log n\right)$
MST	$\Omega(n \log n)$	$O(n \log n)$	$\Omega(n \log n)$	$O(n \log n)$

1.2 Stochastic optimization and adaptivity

Stochastic optimization deals with problems involving uncertainty on the input. In the second half of this thesis, we consider a setting with multiple stages of building a feasible solution. Initially, only some information about the probability distribution of the input is available. At each stage, an "item" is chosen to be included in the solution and the precise properties of the item are revealed (or "instantiated") after we commit to selecting the item irrevocably. The goal is to optimize the expected value/cost of the solution. We set up our stochastic model formally in Section 5.2.

Adaptive and non-adaptive policies. A central paradigm in this setting is the notion of *adaptivity*. Knowing the instantiated properties of items selected so far, we can make further decisions based on this information. We call such an approach an *adaptive policy*. Alternatively, we can consider the model where this information is not available and we must make all decisions beforehand. Such an approach is a *non-adaptive policy*. A fundamental question is, what is the benefit of being adaptive? We measure this benefit quantitatively as the ratio of expected values achieved by an optimal adaptive vs. non-adaptive policy (*the adaptivity gap*). A further question is whether a good adaptive or non-adaptive policy can be found efficiently.

Stochastic Knapsack. The first problem analyzed in this fashion was the Stochastic Knapsack, introduced by Brian Dean. The motivation for this problem is in the area of *stochastic scheduling* where a sequence of jobs should be scheduled on a machine within a limited amount of time. The goal is to maximize the expected profit received for jobs completed before a given deadline. The jobs are processed one by one; after a job has been completed, its precise running time is revealed - but then it is too late to remove the job from the schedule. Hence the property of *irrevocable decisions*, which is essential in the definition of our stochastic model. In joint work with Brian Dean and Michel Goemans, we showed the following results [15].

- Adaptivity can provide a certain benefit which is, however, bounded by a constant factor. A non-adaptive solution which achieves expected value at least 1/7 of the adaptive optimum is achieved by a greedy algorithm which runs in polynomial time.

- For any $\epsilon > 0$, there is a polynomial-time adaptive policy achieving at least $1/5 - \epsilon$ of the adaptive optimum.

- It is PSPACE-hard to answer certain questions concerning adaptive policies, for example:
  "Is it possible to fill the knapsack exactly to its capacity with probability at least p?"

In Chapter 6, we improve the adaptivity gap for Stochastic Knapsack from 7 to 4, by exhibiting a stronger bound on the adaptive optimum and a more efficient *greedy algorithm*. We also describe the PSPACE-hardness results in Chapter 9. For more details on Stochastic Knapsack, we refer to Brian Dean's Ph.D. thesis [14].

Stochastic Packing and Covering. *Stochastic Packing and Covering* problems generalize the deterministic notion of *packing and covering integer programs*. These are linear programs of two types, $Ax \leq b$ or $Ax \geq b$, where $x \in \{0,1\}^n$ is sought to maximize/minimize a linear objective function. These problems can also be viewed as multidimensional knapsack (or knapsack covering) problems, where items have vector sizes with multiple components. Many combinatorial problems fall within this class: e.g. Set Packing, Set Cover, b-matching and general Packing/Covering Integer Programs (PIP/CIP, see [39]). In the stochastic variants of these problems we consider items with random vector sizes which are instantiated upon inclusion of an item in the solution. We consider variants of both packing and covering problems: Set Packing, b-matching, Restricted Packing, Set Cover, Set Cover with item multiplicity, etc. We give the definitions in Section 5.2. Packing problems are then discussed in Chapter 7 and covering problems in Chapter 8.

The analysis of different problem variants reveals a curious pattern of results. Let us present it on the example of Stochastic Set Packing. Here, each item is defined by a value and a probability distribution over subsets $A \subseteq X$ where X is a ground set of cardinality $|X| = d$. A feasible solution is a collection of disjoint sets. It is

known that for deterministic Set Packing, the greedy algorithm provides an $O(\sqrt{d})$-approximation, and there is a closely matching inapproximability result which states that for any fixed $\epsilon > 0$, a polynomial-time $d^{1/2-\epsilon}$-approximation algorithm would imply $NP = ZPP$ [10]. For Stochastic Set Packing, it turns out that:

- The adaptivity gap can be $\Omega(\sqrt{d})$ (by an example inspired by the birthday paradox).
- The adaptivity gap is *bounded* by $O(\sqrt{d})$ (by analyzing an LP bound on the adaptive optimum).
- The proof using an LP leads to a polynomial-time algorithm to find a *fixed set of items* (i.e., a non-adaptive policy) approximating the *adaptive optimum* within a factor of $O(\sqrt{d})$.
- This approximation factor is optimal even if we want to approximate the *non-adaptive optimum* (due to the $d^{1/2-\epsilon}$-inapproximability result).

This phenomenon appears repeatedly: for Set Packing, b-matching, Set Cover and others. For example, the best approximation factor for Set Cover with item multiplicity is $O(\log d)$, the adaptivity gap for Set Cover can be $\Omega(\log d)$ and yet we can approximate the adaptive optimum by an efficient non-adaptive policy within a factor of $O(\log d)$. In general, we design approximation algorithms for most stochastic variants which are near-optimal even in the deterministic cases.

These results hint at a deeper underlying connection between the quantities we are investigating: deterministic approximability, adaptivity gap and stochastic approximability. Note that there is no reference to computational efficiency in the notion of adaptivity gap, so a direct connection with the approximability factor would be quite surprising. Another related quantity is the integrality gap of the associated linear program: in the cases in question, the integrality gap has the same asymptotic behavior as well. In Section 5.3, we establish a connection between the integrality gap and the *randomness gap* for the respective stochastic problem: this is the possible gap between two instances which differ in the probability distributions of item sizes, but not in the expectations of item sizes. The randomness gap is another bound on stochastic approximability (even by means of an adaptive policy) assuming that expectation is the only information available on the item sizes.

Hardness of approximation. We improve some of the previous results on the approximability of PIP. For general Packing Integer Programs, it was known that a polynomial-time $d^{1/2-\epsilon}$-approximation algorithm for any fixed $\epsilon > 0$ would imply $NP = ZPP$ [10]. We improve this negative result to $d^{1-\epsilon}$. It was also known that a $d^{1/(B+1)-\epsilon}$-approximation for PIP with $A \in [0,1]^{d \times n}$ and $\vec{b} = (B, B, \ldots, B)$ ("Restricted Packing") would imply $NP = ZPP$ [10]. We improve this inapproximability factor to $d^{1/B-\epsilon}$.

Summary. Let's summarize the results for Stochastic Packing and Covering. The packing results are presented in Chapter 7 and the covering results in Chapter 8. The hardness results were known except for the general case and Restricted Packing. The integrality gaps were known except for b-matching and Restricted Packing.

PROBLEM VARIANT	Hardness of approx.	Integrality gap	Randomness gap	Adaptivity gap	Stochastic approx.
Packing (general case)	$d^{1-\epsilon}$	$\Theta(d)$	$\Theta(d)$	$\Omega(\sqrt{d}), O(d)$	$O(d)$
Set Packing	$d^{1/2-\epsilon}$	$\Theta(\sqrt{d})$	$\Theta(\sqrt{d})$	$\Theta(\sqrt{d})$	$O(\sqrt{d})$
Restricted Packing	$d^{\frac{1}{B}-\epsilon}$	$\Theta\left(d^{\frac{1}{B}}\right)$	$\Theta\left(d^{\frac{1}{B}}\right)$	$\Omega\left(d^{\frac{1}{B+1}}\right)$	$O\left(d^{\frac{1}{B}}\right)$
B-matching	$d^{\frac{1}{B+1}-\epsilon}$	$\Theta\left(d^{\frac{1}{B+1}}\right)$	$\Theta\left(d^{\frac{1}{B+1}}\right)$	$\Theta\left(d^{\frac{1}{B+1}}\right)$	$O\left(d^{\frac{1}{B+1}}\right)$
Covering (general case)	$\ln d$	∞	∞	$\Omega(d)$	N/A
Set Cover	$\ln d$	$\Theta(\log d)$	$\Omega(d)$	$\Omega(d), O(d^2)$	$O(d)$
Covering +multiplicity	$\ln d$	$\Theta(\log d)$	$\Theta(\log d)$	$\Theta(\log d)$	$O(\log d)$
Set Cover +multiplicity	$\ln d$	$\Theta(\log d)$	$\Theta(\log d)$	$\Theta(\log d)$	$O(\log d)$

Complexity of stochastic optimization with adaptivity. Finally, we show that our class of stochastic optimization problems is closely related to PSPACE. Many natural questions regarding adaptive policies turn out to be PSPACE-hard. The reductions are based on the similarity between adaptive policies and Arthur-Merlin games. For instance, we prove the following results. For details, see Chapter 9.

Theorem 6. *For a knapsack instance with n items, let $\hat{p}$ be the maximum probability that an adaptive policy fills the knapsack exactly to its capacity. Then it is PSPACE-hard to distinguish whether $\hat{p} = 1$ or $\hat{p} \leq 1/2^{n^{1-\epsilon}}$.*

Theorem 7. *For a 2-dimensional Stochastic Knapsack instance, it is PSPACE-hard to maximize the expected value achieved by an adaptive policy.*

Chapter 2

Optimization problems on random subgraphs

In a variety of optimization settings, one has to repeatedly solve instances of the same problem in which only part of the input is changing. It is beneficial in such cases to perform a precomputation that involves only the static part of the input and possibly assumptions on the dynamic part. Our goal is to speed up the subsequent solution of instances arriving at random. The precomputation could possibly be computationally intensive.

In telecommunication networks for example, the topology may be considered fixed but the demands of a given customer (in a network provisioning problem) may vary over time. The goal is to exploit the topology without knowing the demands. The same situation happens in performing multicast in telecommunication networks; we need to solve a minimum spanning tree or Steiner tree problem to connect a group of users, but the topology or graph does not change when connecting different groups of users. In flight reservation systems, departure and arrival locations and times change for each request but schedules do not (availability and prices do change as well but on a less frequent basis). Yet another example is for delivery companies; they have to solve daily vehicle routing problems in which the road network does not change but the locations of customers to serve do.

Examples of such *repetitive* optimization problems with both static and dynamic inputs are countless and in many cases it is unclear if one could take any advantage of the advance knowledge of the static part of the input. One situation which has been much studied (especially from a practical point-of-view) is for s-t shortest path queries in large-scale navigation systems or Geographic Information Systems. In that setting, it is too slow to compute from scratch the shortest path whenever a query comes in. Various preprocessing steps have been proposed, often creating a hierarchical view of the network, see for example [26].

We study several combinatorial problems for which such precomputation might be useful. First, we consider the minimum spanning tree (MST) problem, which is one of the basic optimization problems and it also serves as a building block in more sophisticated algorithms and heuristics. We would like to find a sparse subgraph Q of a given graph G such that the minimum spanning tree of a random subgraph of G is

contained in Q almost always. This would speed up subsequent random MST queries, by restricting our attention to the edges in Q. Our answer would be guaranteed to be correct with high probability.

In the following chapter, we turn to another problem along these lines where we would like to approximate the shortest-path metric for random subgraphs, by selecting a priori a subgraph Q whose metric approximates that of a random subgraph. This would then allow us to speed up the solution of any combinatorial problem, based on the shortest-path metric, such as Steiner Tree, Facility Location or Traveling Salesman. Again, we assume that instances are generated as random subgraphs.

2.1 The MST-covering problem

The MST-covering problem. Assume we are given an edge-weighted graph $G = (V, E)$ with n vertices and m edges and we would like to (repeatedly) find the minimum-weight spanning tree of either a vertex-induced subgraph $H = G[W]$, $W \subseteq V$ (the *vertex case*) or a subgraph $H = (V, F)$, $F \subseteq E$ (the *edge case*). In general, we need to consider the minimum spanning forest, i.e. the minimum spanning tree on each component, since the subgraph might not be connected. We denote this by $MST(W)$ or $MST(F)$.

Our primary focus is a random setting where each vertex appears in W independently with probability p (in the vertex case; we denote $W = V(p)$); secondly, a setting where each edge appears in F independently with probability p (in the edge case; we denote $F = E(p)$). The question we ask is whether there exists a *sparse set* of edges Q which contains the minimum spanning forest of the random subgraph *with high probability*.

We also address a deterministic setting where we assume that W is obtained from V by removing a fixed number of vertices, or F is obtained from E by removing a fixed number of edges (by a *malicious adversary*). Then we seek a sparse set of edges Q containing the minimum spanning forests of *all such subgraphs*.

More precisely, if the minimum spanning tree is not unique, we ask that Q contains *some* minimum spanning tree. Alternatively, we can break ties by an arbitrary fixed ordering of edges, and require that Q contains the unique minimum spanning tree. This is a stronger requirement and in the following, we will indeed assume that the edge weights are distinct and the minimum spanning trees is unique.

Example. Consider a complete graph G on vertices $V = \{1, 2, \ldots, n\}$ where the weight of edge $(i, j), i < j$ is $w(i, j) = 2^i$. Assume that $W \subseteq V$ is sampled uniformly, each vertex with probability $1/2$. It is easy to see that $MST(W)$ is a star of edges centered at the smallest i in W and connecting i to the remaining vertices in W. The probability that $(i, j) \in MST(W)$ (for $i < j$) is $1/2^{i+1}$ since $\{i, j\}$ must be in W and no vertex smaller than i can be in W. Note that when we order the edges (i, j) lexicographically, their probabilities of appearance in MST(W) decrease exponentially, by a factor of 2 after each block of roughly n edges. If this were always the case, we could take $O(n \log n)$ edges with the largest probability of appearance in

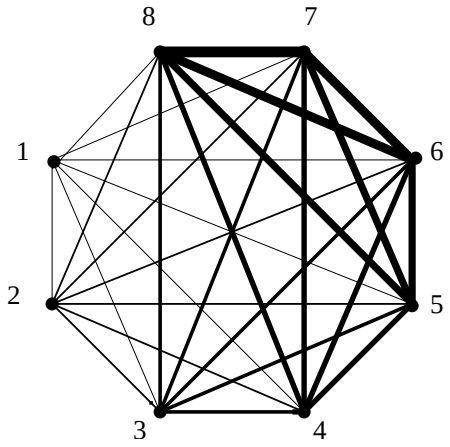

Figure 2-1: A complete graph K_6 with lexicographically ordered edges. The edge weights are marked by thickness.

$MST(W)$ and the probabilities for the remaining edges would be polynomially small which would solve our problem. An example of an MST-covering set here is

$$Q = \{(i,j) \in E : i < 3\log_2 n\};$$

then, any edge in $E \setminus Q$ appears in $MST(W)$ with probability at most $1/n^3$.

Also, our example demonstrates that we need to include $\Omega(n \log n)$ edges in Q if $\Pr[MST(W) \setminus Q \neq \emptyset]$ should be polynomially small. More generally, this is true for any weighted graph - just consider the event that a vertex is isolated in $Q[W]$. Unless Q contains at least $\log_2 n$ edges incident with every vertex, some vertex gets isolated in $Q[W]$ with probability at least $1/n$. Then Q cannot be a good MST-covering set. We will make a more precise argument in Section 3.8 but these examples indicate that $|Q| = O(n \log n)$ is the correct bound to shoot for.

2.2 The metric approximation problem

Metric-approximating edge sets. Given a graph $G = (V, E)$ with edge lengths, we would like to repeatedly solve a metric-based optimization problem such as Shortest Path, Steiner Tree, Facility Location or Traveling Salesman. The common feature of these problems is that the cost of a solution depends only on distances between pairs of vertices in G (and possibly on some cost function associated with vertices). We would like to speed up the solution of such problems, in case each instance is drawn as a random subgraph, either a vertex-induced subgraph $H = G[W],\ W \subseteq V$ (the *vertex case*) or a subgraph $H = (V, F),\ F \subseteq E$ (the *edge case*). For that purpose, we would like to find a sparse set of edges $Q \subseteq E$ such that the shortest-path metric of $Q \cap E(H)$ approximates the shortest-path metric of $E(H)$ within a constant factor c, with high probability. We call such a set *c-metric-approximating*.

Example. In the example of Section 2.1, the MST covering set Q would also be a 1-metric-approximating set, i.e. preserving the metric *exactly* with high probability. This is because the minimum spanning tree of any induced subgraph $G[W]$ is a star which defines the shortest-path metric exactly. However, this phenomenon will turn out to be quite rare and not attainable in general.

2.3 Literature review

2.3.1 MST covering

The absence of complete knowledge of the input data is the defining characteristic of stochastic optimization. Very often, part of the input is random and one has to make a decision in the first stage without knowing the realization of the stochastic components; these are then revealed and one has to make further decisions. Although the minimum spanning tree problem (as a prototypical combinatorial optimization problem) has been considered in a wide variety of settings with incomplete or changing data, it has not been under the particular viewpoint considered here.

In dynamic graph algorithms, one assumes that the graph is dynamically changing and one needs to update the solution of the problem after each input update. For a minimum spanning tree problem in which edges can be inserted or deleted, the best known dynamic algorithm has amortized cost $O(\log^4 n)$ per operation [24]. This is not efficient here though, since our instances are changing too drastically.

In the NP-hard Probabilistic MST problem [4], each vertex becomes a "terminal" independently with a given probability. However, the goal here is to find a spanning tree such that the Steiner tree obtained by removing the edges not needed to connect the terminals has minimum expected cost. Our different model has the advantage of giving a minimum spanning tree (instead of a suboptimal spanning tree) at the expense of a logarithmic increase in the number of edges.

In practice, graph optimization problems are often solved on a sparse subgraph, and edges which are not included are then *priced* to see if they could potentially improve the solution found, see for example [1] for the matching problem. Our results can therefore be viewed as a theoretical basis for this practice in the case of the MST, and give precise bounds on the sparsity required.

2.3.2 Metric approximation

Sparse subgraphs approximating the shortest-path metric have been considered in ample scope, under various restrictions - for geometric graphs, general graphs, with constrained degrees, restricted structure, etc. Metric-approximating subgraphs are known as *spanners*. We say that $S \subset G$ is a c-spanner if for any path in G of length l there is a corresponding path in S of length at most cl. See for example [13] for a survey of results about general spanners.

The existence of spanners with a low number of edges is related to the existence of graphs without short cycles. We say the graph G has *girth* g, if the length of the

shortest cycle is g. A c-spanner for a graph of girth $g \geq c+2$ (with unit edge lengths) must be the graph itself, since no edge can be replaced by another path of length at most c. Therefore the number of edges required in general for a c-spanner is at least the extremal number of edges for a graph of girth $g = c + 2$. It is known that there are graphs of girth g with $n^{1+1/(g-1)}$ edges (in classical work by Paul Erdős [35]; the proof uses the probabilistic method).

On the other hand, a spanner can be constructed by a construction avoiding cycles shorter than $c + 1$, which yields c-spanner of girth at least $c + 2$ (see [13]). Thus the extremal number of edges for graphs of given girth provides an upper bound on the size of spanners as well. The best known upper bound on the number of edges for a graph of girth g is $O(n^{1+2/(g-2)})$ for $g \geq 4$ even [3]. So there always exists a c-spanner with $O(n^{1+2/c})$ edges.

However, our requirements are stronger than those for a spanner: we ask that our subgraph Q approximates the metric of $G[W]$ even when Q is restricted to the subgraph induced by W. In other words, we are not allowed to use paths in Q leaving the subset W.

Chapter 3

Covering the minimum spanning tree of random subgraphs

3.1 The first attempt

Let's start with the variant of the MST covering problem where vertices are removed at random. For now, assume that each vertex is removed independently with probability $1/2$ and denote the set of surviving vertices by W. In this section, we are not going to achieve the optimal upper bound, but we develop a simple algorithm which demonstrates that at least some non-trivial upper bound exists. Naturally, taking all $O(n^2)$ edges of the graph for Q is a valid solution, but we would like to improve significantly on this.

The intuitive idea behind this algorithm is that of "path covering". This is similar to the usual procedure to construct a spanner. Note that in building a minimum spanning tree (consider for example Prim's algorithm), an edge (u, v) is not used if there is a $u - v$ path containing only edges of smaller weight. In such a case, we say that a *path covers* (u, v). This does not mean that (u, v) cannot appear in the MST of a subgraph of G, but it is an indication that (u, v) might not be necessary for our covering set Q. Let's describe our first algorithm now.

Algorithm 1. (given parameters $c, r \in \mathbb{Z}_+$)

1. Let $E_1 := E(G)$. Repeat the following for $k = 1, 2, \ldots, r$.
2. Let $Q_k := \emptyset$.
3. Process the edges of E_k in the order of increasing edge weights.
4. For each edge $(u, v) \in E_k$, unless it is covered by a path of length at most c in Q_k, include (u, v) in Q_k.
5. If $k < r$, set $E_{k+1} := E_k \setminus Q_k$ and go back to 2.
6. Finally, let $Q := \cup_{k=1}^{r} Q_k$.

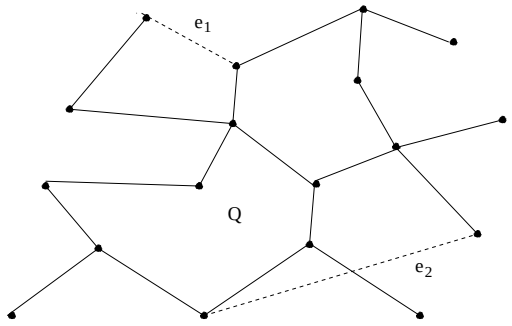

Figure 3-1: An example of path covering. The bold edges are already included in Q. The dashed edges are considered for inclusion; edge e_1 is covered by a path of length 3, while the edge e_2 is covered only by a path of length 4.

Note that in each stage k, this algorithm maintains two useful properties. Any edge which is *not* in Q_k is covered by a path of length at most c; this path uses edges of smaller weight, which serves to decrease the probability that $(u, v) \in MST(W)$. Secondly, the algorithm avoids all cycles shorter than $c + 2$ to be created in Q_k; this serves to bound the size of Q.

Lemma 1. *For $c = 3$ and $r = \lfloor 12 \ln n \rfloor$, Algorithm 1 finds in polynomial time a set Q of size $|Q| \leq O(n^{3/2} \ln n)$ such that for $W \subseteq V$ sampled uniformly at random,* $\Pr[MST(W) \subseteq Q] > 1 - 1/n$.

Proof. For each $k = 1, 2, \ldots, r$, Q_k is a set of edges avoiding a C_4. By a well-known result in extremal graph theory [3], $|Q_k| = O(n^{3/2})$. Therefore $|Q| = O(rn^{3/2}) = O(n^{3/2} \log n)$.

Now consider an edge $(u, v) \in E$ that we have not chosen in any stage, i.e. $(u, v) \notin \cup_{k=1}^{r} Q_k$. For each k, there is a covering $u - v$ path of length 2 or 3. By construction, these paths are edge-disjoint. For a path of length 2, there is probability $1/2$ that it survives in $G[W]$. For a path of length 3, the probability is $1/4$.

Note that if two of these paths share a vertex w, then they must be both of length 3 and w is a common neighbor of u and v. In that case, we remove the two paths and replace them by a path of length 2, $u-w-v$. For two vertex-disjoint paths of length 3, the probability that neither of them survives in $G[W]$ is $(1-1/4)^2$ which is more than the probability of destruction for a single path of length 2. Therefore we may assume that we have r vertex-disjoint $u - v$-covering paths of length 3; $(u, v) \in MST(W)$ can occur only if $u, v \in W$ while none of these paths survive in W:

$$\Pr[(u, v) \in MST(W)] \leq \frac{1}{4}\left(1 - \frac{1}{4}\right)^r < \frac{1}{4}e^{-r/4} \leq \frac{1}{n^3},$$

$$\Pr[\exists (u, v) \in E \setminus Q; (u, v) \in MST(W)] < \frac{1}{n}.$$

□

One might try to strengthen the argument by checking for paths longer than 3

but in the case of vertex-sampling, $c > 3$ creates difficulties because of the positive correlation among overlapping paths. Still, this idea works in the edge-sampling case. Then we choose c optimally to cover edges by paths of length at most $\sqrt{\ln n}$ which yields a near-linear bound on Q.

Lemma 2. *For $c = \lfloor\sqrt{\ln n}\rfloor$ and $r = \lfloor 2^{c+2} \ln n\rfloor$, Algorithm 1 finds in polynomial time a set Q of size $O\left(n\, e^{3\sqrt{\ln n}}\right)$ such that for $F \subseteq E$ sampled uniformly at random, $\Pr[MST(F) \subseteq Q] > 1 - 1/n^2$.*

Proof. To estimate the size of Q, note that each Q_k is a set of edges avoiding cycles of length smaller than $c+2$, which has size $O(n^{1+2/c})$ (see [3]). Thus

$$|Q| = O(rn^{1+2/c}) = O(2^c n^{1+2/c} \log n) = O(n\, e^{3\sqrt{\ln n}}).$$

To prove that Q is a good covering set, note that any $(u, v) \in E \setminus Q$ is covered by r *edge-disjoint* paths of length at most c. Each of them has probability at least $1/2^c$ of survival and $(u, v) \in MST(F)$ is possible only if none of the r paths survive. Here, the events are independent because of edge-disjointness.

$$\Pr[(u, v) \in MST(F)] \leq \frac{1}{2}\left(1 - \frac{1}{2^c}\right)^r < \frac{1}{2}e^{-r/2^c} \leq \frac{1}{n^4}.$$

By the union bound over all edges, $MST(F) \subseteq Q$ with probability at least $1 - 1/n^2$. □

Thus we have bounds $O(n^{3/2} \log n)$ for the vertex case and $O(n\, e^{3\sqrt{\ln n}})$ for the edge case, respectively. These are actually the best bounds we are able to achieve by efficient deterministic algorithms. Also, note that the covering paths here are guaranteed to have short length. In fact, Q is also a 3-metric-approximating set. We explore this idea further in Chapter 4. In the following, we turn to probabilistic methods which lead to the asymptotically optimal bound of $O(n \log n)$ for the MST covering problem.

3.2 The boosting lemma

We start by analyzing the event that a fixed edge appears in the minimum spanning tree of a random induced subgraph. We would like to show that the probability of this event cannot be too high for too many edges.

The property of belonging to the minimum spanning tree of an induced subgraph has a very simple property, and that is *down-monotonicity*: intuitively, removing more vertices makes an edge only more likely to appear in the MST.

Lemma 3. *For an edge $(u, v) \in E$, let $X = V \setminus \{u, v\}$ and let $\mathcal{F}$ denote the family of vertex sets $A \subseteq X$ for which (u, v) is in the minimum spanning forest of the induced subgraph $G[A \cup \{u, v\}]$. Then $\mathcal{F}$ is a down-monotone family:*

$$A \in \mathcal{F}, B \subseteq A \Longrightarrow B \in \mathcal{F}.$$

Proof. It is easy to see (for example, from Prim's algorithm) that $(u,v) \in MST(A \cup \{u,v\})$ if and only if there is no $u-v$ path in $G[A \cup \{u,v\}]$ using only edges of smaller weight than (u,v). If there is no such path in $G[A \cup \{u,v\}]$ then there is no such path in $G[B \cup \{u,v\}]$ either. □

For a random $A \subseteq X$, we say that "$A \in \mathcal{F}$" is a *down-monotone event*. Knowing that the appearance of an edge in the MST of a random subgraph is a down-monotone event will be useful in the following sense: if an edge appears in $MST(W)$ with probability ϵ, the probability that it appears in $MST(S)$, where $S \subseteq V$ is a random set substantially smaller than W, will be much higher than ϵ. This will allow us to bound the number of such edges, because not too many edges can appear in a minimum spanning tree with a large probability.

We prove a general inequality for down-monotone events. We call this inequality the *boosting lemma*, since it states how the probability of a down-monotone event is boosted, when we decrease the sampling probability.

Lemma 4 (The Boosting Lemma)**.** *Let X be a finite set and $\mathcal{F} \subseteq 2^X$ a down-monotone family of subsets of X. Let $\vec{p} \in [0,1]^n$ and sample a random subset $X(\vec{p})$ by choosing element i independently with probability p_i. Define*

$$\sigma_p = \Pr[X(\vec{p}) \in \mathcal{F}].$$

Let $\gamma \in (0,1)$ and similarly define $\sigma_q = \Pr[X(\vec{q}) \in \mathcal{F}]$ where element i is sampled with probability $q_i = 1-(1-p_i)^\gamma$. Then

$$\sigma_q \geq \sigma_p^\gamma.$$

Proof. We proceed by induction on $|X|$. For $X = \emptyset$ the statement is trivial ($\sigma_p = \sigma_q = 0$ or 1). Let $a \in X$, $Y = X \setminus \{a\}$ and define

- $\mathcal{F}_0 = \{A \subseteq Y : A \in \mathcal{F}\}$
- $\mathcal{F}_1 = \{A \subseteq Y : A \cup \{a\} \in \mathcal{F}\}$

By down-monotonicity, we have $\mathcal{F}_1 \subseteq \mathcal{F}_0$. Next, we express σ_q by the law of conditional probabilities:

$$\sigma_q = \Pr[X(\vec{q}) \in \mathcal{F}] = q_a \Pr[Y(\vec{q}) \in \mathcal{F}_1] + (1-q_a)\Pr[Y(\vec{q}) \in \mathcal{F}_0]$$

where $Y(\vec{q})$ denotes a subset of Y sampled with the respective probabilities q_i; $Y(\vec{q}) = X(\vec{q}) \setminus \{a\}$. We denote $\omega_p = \Pr[Y(\vec{p}) \in \mathcal{F}_1]$ and $\tau_p = \Pr[Y(\vec{p}) \in \mathcal{F}_0]$. By induction, we can apply the boosting lemma to the down-monotone events $\mathcal{F}_0, \mathcal{F}_1 \subseteq 2^Y$: $\omega_q \geq \omega_p^\gamma, \tau_q \geq \tau_p^\gamma$. We get

$$\sigma_q = q_a\omega_q + (1-q_a)\tau_q \geq (1-(1-p_a)^\gamma)\omega_p^\gamma + (1-p_a)^\gamma\tau_p^\gamma.$$

Note that $\omega_p \leq \tau_p$ because $\mathcal{F}_1 \subseteq \mathcal{F}_0$. It remains to prove the following:

$$(1-(1-p)^\gamma)\omega^\gamma + (1-p)^\gamma\tau^\gamma \geq (p\omega + (1-p)\tau)^\gamma \tag{3.1}$$

for any $p \in [0,1], \gamma \in (0,1), 0 \leq \omega \leq \tau$. Then we would conclude that

$$\sigma_q \geq (p_a \omega_p + (1-p_a)\tau_p)^\gamma = \sigma_p^\gamma$$

using the law of conditional probabilities for $\sigma_p = \Pr[X(\vec{p}) \in \mathcal{F}]$.

We verify Equation 3.1 by analyzing the difference of the two sides: $\phi(\tau) = (1-(1-p)^\gamma)\omega^\gamma + (1-p)^\gamma\tau^\gamma - (p\omega + (1-p)\tau)^\gamma$. We show that $\phi(t) \geq 0$ for $t \geq \omega$. For $t = \omega$, we have $\phi(t) = 0$. By differentiation,

$$\begin{aligned}\phi'(t) &= \gamma(1-p)^\gamma t^{\gamma-1} - \gamma(1-p)(p\omega + (1-p)t)^{\gamma-1} \\ &= \gamma(1-p)^\gamma t^{\gamma-1}\left(1 - \left(\frac{(1-p)t}{p\omega + (1-p)t}\right)^{1-\gamma}\right) \geq 0.\end{aligned}$$

Therefore $\phi(t) \geq 0$ for any $t \geq \omega$ which completes the proof. □

Note. The boosting lemma is tight for $\mathcal{F} = 2^A, A \subset X$ in which case $\sigma_p = (1-p)^{|X \setminus A|}$. This form of the lemma is the most general we have found; more restricted versions are easier to prove. For probabilities p_i, q_i satisfying $(1-p_i) = (1-q_i)^k$, $k \in \mathbb{Z}$, there is a simple probabilistic proof using repeated sampling:

Sample independently subsets $Y_j = X(\vec{q}), j = 1, 2, \ldots, k$, and set $Y = Y_1 \cup Y_2 \cup \ldots \cup Y_k$. Element i has probability $(1-q_i)^k = (1-p_i)$ that it does not appear in any Y_j, therefore Y is effectively sampled with probabilities p_i. Then we get, from the monotonicity of $\mathcal{F}$:

$$\sigma_p = \Pr[Y \in \mathcal{F}] \leq \Pr[\forall j; Y_j \in \mathcal{F}] = \sigma_q^k.$$

Another special case is when the sampling probability p is the same for all elements and $\sigma_p = (1-p)^k$ for some $k \in \mathbb{Z}$. Then we can argue by the Kruskal-Katona theorem that $\mathcal{F}$ contains at least $\binom{n-k}{a}$ subsets of size a for any $a \leq n-k$, which implies that $\sigma_q \geq (1-q)^k$ for any $q \leq p$. It is not clear if this applies to $\sigma_p = (1-p)^k$ for non-integer k.

In [6], Bollobás and Thomason prove a lemma about down-monotone events which applies to random subsets of fixed size: If P_r is the probability that a random subset of size r is in $\mathcal{F}$, then for any $s \leq r$,

$$P_s^r \geq P_r^s.$$

Considering that the two random models are roughly equivalent (instead of sampling with probability p, take a random subset of size pn), this lemma has a very similar flavor to ours. However, putting the two random models side by side like this, the Bollobás-Thomason lemma is weaker; for example, compare $p = 1 - 1/n, q = 1/2$ and $r = n-1, s = n/2$. Our boosting lemma implies $\sigma_q \geq (\sigma_p)^{1/\log n}$. The Bollobás-Thomason lemma says only $P_s \geq \sqrt{P_r}$.

3.3 Covering the MSTs of random vertex-induced subgraphs

Now we prove the bound on the size of covering sets for the case of randomly sampled vertices. As noted before, for any edge $(u, v) \in E$, the event that $(u, v) \in MST(W)$, conditioned on $u, v \in W$, is down-monotone. Let's denote

$$\sigma_p(u, v) = Pr_W[(u, v) \in MST(W) \mid u, v \in W]$$

where $W = V(p)$ contains each vertex independently with probability p.

Lemma 5. *For a weighted graph G on n vertices, $0 < p < 1$, and $k \geq 1/p$, let*

$$Q_k^{(p)} = \{(u, v) \in E : \sigma_p(u, v) \geq (1 - p)^{k-1}\}.$$

Then

$$|Q_k^{(p)}| < ekn.$$

Proof. Sample a random subset $S = V(q)$, with probability $q = 1/k \leq p$. For every edge $(u, v) \in Q_k^{(p)}$, we have $\sigma_p(u, v) \geq (1-p)^{k-1}$, and therefore by the boosting lemma with $\gamma = \frac{\ln(1-q)}{\ln(1-p)}$, we get $\sigma_q(u, v) \geq (1 - q)^{k-1}$. Consequently, $Q_k^{(p)} \subseteq Q_k^{(q)}$ and

$$Pr[(u, v) \in MST(S)] = q^2 \sigma_q(u, v) \geq \frac{1}{k^2} (1 - q)^{k-1} > \frac{1}{ek^2},$$

$$\mathbf{E}[|MST(S)|] \geq \sum_{(u,v) \in Q_k^{(p)}} Pr[(u, v) \in MST(S)] > \frac{|Q_k^{(p)}|}{ek^2}.$$

On the other hand, the size of the minimum spanning forest on S is at most the size of S, and so

$$\mathbf{E}[|MST(S)|] \leq \mathbf{E}[|S|] = \frac{n}{k}.$$

Combining these two inequalities, we get $|Q_k^{(p)}| < ekn$. □

Theorem 8. *Let G be a weighted graph on n vertices, $0 < p < 1$, and $c > 0$. Let $b = 1/(1 - p)$. Then there exists a set $Q \subseteq E$ of size*

$$|Q| \leq e(c + 1)n \log_b n + O(n)$$

such that for a random $W = V(p)$,

$$Pr_W[MST(W) \subseteq Q] > 1 - \frac{1}{n^c}.$$

Proof. Order the edges by the decreasing values of $\sigma_p(u, v)$. Partition the sequence into blocks $B_1, B_2, \ldots \subset E$ of size $\lceil en \rceil$. Lemma 5 implies that for any $(u, v) \in B_{k+1}$, $k \geq 1/p$,

$$Pr[(u, v) \in MST(W)] = p^2 \ \sigma_p(u, v) < p^2(1 - p)^{k-1}.$$

Define Q as the union of the first $h = \lceil (c+1)\log_b n \rceil + 2$ blocks, $Q = \bigcup_{k=1}^{h} B_k$. We have $h \geq 1/p$ (for $p \geq 1/2$ it's obvious that $h \geq 2 \geq 1/p$, and for $p < 1/2$, $h > \log_b n > \frac{\ln n}{2p} > 1/p$ for $n > e^2$). So we can apply the above bound to blocks starting from $h+1$:

$$\begin{aligned} Pr[MST(W) \setminus Q \neq \emptyset] &\leq \sum_{(u,v)\in E\setminus Q} Pr[(u,v) \in MST(W)] \\ \leq \sum_{k=h+1}^{\infty} \lceil en \rceil p^2 (1-p)^{k-2} &= \lceil en \rceil p (1-p)^{h-1} < \frac{\lceil en \rceil p(1-p)}{n^{c+1}} < \frac{1}{n^c}. \end{aligned}$$

□

3.4 Improving the constant factor

In the results of Lemma 5 and Theorem 8, we get a factor of e which however seems to be an artifact of the probabilistic proof. In fact it is possible that the best upper bound has a constant factor of 1. The example in Section 3.8 shows that this would be tight. In this section, we show that e is indeed just a product of the proof and we prove a somewhat improved constant factor.

Lemma 6. *For the set $Q_k^{(p)}$ from Lemma 5 and any $k \geq 1/p$,*

$$|Q_k^{(p)}| < (1 + e/2)kn.$$

Proof. Consider more carefully the argument bounding $|Q_k^{(p)}|$:

$$pn \geq \mathbf{E}[|MST(V(p))|] \geq \sum_{l=1}^{\infty} |Q_l^{(p)} \setminus Q_{l-1}^{(p)}|\ p^2(1-p)^{l-1} = \sum_{l=1}^{\infty} |Q_l^{(p)}|\ p^3(1-p)^{l-1},$$

i.e.

$$n \geq \sum_{l=1}^{\infty} |Q_l^{(p)}|\ p^2(1-p)^{l-1}.$$

Our previous argument basically replaces all the terms for $l < k$ by zero, which leads to the factor of e (for $p = 1/k$). However, we can get a better upper bound, if we can *lower bound* these terms as well. And that is certainly possible - let's assume that G is a complete graph (without loss of generality for the upper bound). Then $Q_l^{(p)}$ must be l-vertex-connected, otherwise there is an edge $(u,v) \in E \setminus Q_l^{(p)}$ which is not covered by l vertex-disjoint paths in $Q_l^{(p)}$ which would imply $\sigma_p(u,v) \geq (1-p)^l$. Therefore $|Q_l^{(p)}| \geq ln/2$.

More generally, assume that

- $\forall l < k;\ |Q_l^{(p)}| \geq \gamma ln$, for some constant $\gamma > 0$.

Then we get, replacing $|Q_l^{(p)}|$ by γln for $l < k$ and by $|Q_k^{(p)}|$ for $l > k$,

$$\begin{aligned} n &\geq \sum_{l=1}^{k-1} \gamma l n p^2 (1-p)^{l-1} + \sum_{l=k}^{\infty} |Q_k^{(p)}| p^2 (1-p)^{l-1} \\ &= \gamma n \sum_{l=1}^{k-1} \sum_{j=l}^{k-1} p^2 (1-p)^{j-1} + |Q_k^{(p)}|\ p(1-p)^{k-1} \\ &= \gamma n \sum_{l=1}^{k-1} p((1-p)^{l-1} - (1-p)^{k-1}) + |Q_k^{(p)}|\ p(1-p)^{k-1} \\ &= \gamma n (1 - (1-p)^{k-1}(1 + p(k-1))) + |Q_k^{(p)}|\ p(1-p)^{k-1}. \end{aligned}$$

This implies a bound on $|Q_k^{(p)}|$, namely

$$|Q_k^{(p)}| \leq \left(\frac{1-\gamma}{(1-p)^{k-1}} + \gamma(1 + p(k-1)) \right) \frac{n}{p}. \tag{3.2}$$

Using the l-connectivity argument, we can use $\gamma = 1/2$, and for $q = 1/k$ we get

$$|Q_k^{(q)}| \leq \left(\frac{1/2}{(1-q)^{k-1}} + 1 \right) \frac{n}{q} \leq \left(\frac{e}{2} + 1 \right) kn.$$

For $p \geq q$, we apply the boosting lemma which as in the proof of Lemma 5 implies that $Q_k^{(p)} \subseteq Q_k^{(q)}$. Therefore the bound holds for any $p \geq 1/k$ as well. □

As a direct consequence, we get an improved version of Theorem 8.

Theorem 9. *Let G be a weighted graph on n vertices, $0 < p < 1$, and $c > 0$. Let $b = 1/(1-p)$. Then there exists a set $Q \subseteq E$ of size*

$$|Q| \leq (1 + e/2)(c+1) n \log_b n + O(n)$$

such that for a random $W = V(p)$,

$$Pr_W[MST(W) \subseteq Q] > 1 - \frac{1}{n^c}.$$

3.5 Covering the MSTs of subgraphs of fixed size

Directly from Lemma 6, we also get the following interesting implication for the "deterministic version" of the problem, where at most $k-1$ vertices can be removed arbitrarily.

Corollary 1. *For any weighted graph G on n vertices, and $k \in \mathbb{Z}_+$, there exists a set $Q_k \subseteq E$ of size*

$$|Q_k| < (1 + e/2)kn$$

which contains $MST(W)$ for any $|W| > n - k$.

Proof. Let $Q_k = \bigcup_{|W|>n-k} MST(W)$. For $k = 1$ the lemma is trivial (Q_1 is the minimum spanning tree of G). For $k \geq 2$, choose $p = 1/2$ and consider $Q_k^{(p)}$ as defined in Lemma 5. Lemma 6 which states that $|Q_k^{(p)}| < (1 + e/2)kn$. For any edge (u, v) which appears in $MST(W)$ for $|W| > n - k$, $\sigma_p(u, v) \geq (1 - p)^{k-1}$, since the vertices in $V \setminus W$ are removed with probability at least $(1 - p)^{k-1}$; therefore $Q_k \subseteq Q_k^{(p)}$. □

Observe that the set Q_k can be found in polynomial time. For every edge (u, v), its membership in Q_k can be tested by computing the vertex connectivity between vertices u, v in the subgraph of edges lighter than (u, v). By Menger's theorem, $(u, v) \in Q_k$ if and only if there are no k vertex-disjoint $u - v$ paths using edges of smaller weight than (u, v). This, however, does not seem to imply a bound on the size of Q_k easily. The only way we can prove our bound is through probabilistic reasoning.

It is not difficult to see that $|Q_1| \leq n - 1$ and $|Q_2| \leq 2n - 3$. It is also possible to define edge weights so that Q_k must contain $(n-1)+(n-2)+\cdots+(n-k) = kn-\binom{k+1}{2}$ edges (see Section 3.8 for an example). We conjecture this to be the actual tight upper bound. Similarly, we conjecture that $kn - \binom{k+1}{2}$ is the best possible bound on $|Q_k^{(p)}|$ in Lemma 5 (and this would be also achieved for the graph described in Section 3.8).

The same question in the edge case is easier to answer. The number of edges in all MSTs obtained upon removing at most $k - 1$ edges can be bounded by $k(n - 1)$, by finding the minimum spanning tree and removing it from the graph, repeatedly k times. (Which also works for multigraphs, and more generally for matroids.) For simple graphs, we can prove a bound of $kn - \binom{k+1}{2}$ which is tight (see the examples constructed in Section 3.8).

Lemma 7. *For any (simple) weighted graph on n vertices and $1 \leq k \leq n$, there exists a set $Q_k \subseteq E$ of size*

$$|Q_k| \leq \sum_{i=1}^{k} (n - i) = kn - \binom{k+1}{2}$$

which contains the minimum spanning forest $MST(F)$ for any $|F| > m - k$.

Proof. Let $Q_k = \bigcup_{|F|>m-k} MST(F)$. We proceed by induction on n. For $n = k$, it is trivial that $|Q_k| \leq \binom{n}{2}$. So assume $n > k$.

Consider the heaviest edge $e^* \in Q_k$. Since $e^* \in MST(F)$ for some $|F| > m - k$, there is a cut $\delta(V_1) = \{(u, v) \in E : u \in V_1, v \notin V_1\}$ such that e^* is the *lightest* edge in $\delta(V_1) \cap F$. Consequently $Q_k \cap \delta(V_1) \subseteq (E \setminus F) \cup \{e^*\}$, which means that at most k edges of Q_k are in the cut $\delta(V_1)$. Let $V_2 = V \setminus V_1$ and apply the inductive hypothesis on $G_1 = G[V_1]$ and $G_2 = G[V_2]$, and their respective MST-covering sets $Q_{k,1}, Q_{k,2}$. We use the following characterization of Q_k: $(u, v) \in Q_k \Leftrightarrow$ there are no k edge-disjoint $u - v$ paths in the subgraph of edges lighter than (u, v) (again by Menger's theorem, for edge connectivity). Since the edge connectivity in G is at least as strong as the edge connectivity in G_1 or G_2, it follows that $Q_k[V_1] \subseteq Q_{k,1}$, $Q_k[V_2] \subseteq Q_{k,2}$ and we get

$$|Q_k| \leq |Q_{k,1}| + |Q_{k,2}| + k.$$

Let $n_1 = |V_1|, n_2 = |V_2|$; $n = n_1 + n_2 > k$. We distinguish two cases:

- If one of n_1, n_2 is at least k, assume it is n_1. By the inductive hypothesis, $|Q_{k,1}| \leq \sum_{i=1}^{k} (n_1 - i)$, and $|Q_{k,2}| \leq k(n_2 - 1)$ (which holds for any n_2, smaller or larger than k), so

$$|Q_k| \leq \sum_{i=1}^{k} (n_1 - i) + k(n_2 - 1) + k = \sum_{i=1}^{k} (n - i).$$

- If both $n_1, n_2 < k$, then we estimate simply $|Q_{k,1}| \leq \binom{n_1}{2}$, $|Q_{k,2}| \leq \binom{n_2}{2}$. We get

$$|Q_k| \leq \binom{n_1}{2} + \binom{n_2}{2} + k \leq \frac{k(n_1 - 1)}{2} + \frac{k(n_2 - 1)}{2} + k = \frac{kn}{2} \leq \sum_{i=1}^{k} (n - i).$$

□

3.6 Algorithmic construction of covering sets

It is natural to ask whether the covering sets can be found efficiently. In the deterministic case, we have shown that this is quite straightforward (Section 3.5). For the probabilistic case, we have shown deterministic algorithms in Section 3.1, but the covering sets obtained there are larger than optimal. Here, we would like to find covering sets of size that is implied by the boosting lemma. However, we have to keep in mind that it is not possible to test whether $(u, v) \in Q_k^{(p)}$ directly. This would amount to calculating the *u-v-reliability* in the graph of edges lighter than (u, v), which is a #P-complete problem [33].

Still, we can find a covering set Q with an efficient randomized algorithm, which takes advantage of the boosting lemma as well. It is a Monte Carlo algorithm, in the sense that it finds a correct solution with high probability, but the correctness of the solution cannot be verified easily.

Algorithm 2. Given $G = (V, E)$, $w : E \to \mathbb{R}$, $0 < p < 1$, $c > 0$.

- Let $b = 1/(1 - p)$ and $k = \lceil (c + 2) \log_b n \rceil + 1$.
- Repeat the following for $i = 1, \ldots, r = \lceil 32ek^2 \ln n \rceil$:
  - Sample $S_i \subseteq V$, each vertex independently with probability $q = 1/k$.
  - Find $T_i = MST(S_i)$.
- For each edge, include it in Q if it appears in at least $16 \ln n$ different T_i's.

The running time of Algorithm 2 is determined by the number of calls to an MST procedure, which is $O(\log_b n \ln n)$. Since a minimum spanning forest can be found in time $O(m\alpha(m, n))$ deterministically [9] or $O(m)$ randomized [28], for constant $b = 1/(1 - p)$ we get a running time near-linear in m.

Theorem 10. *Algorithm 2 finds with high probability a set $Q \subseteq E$ such that*

$$|Q| \leq 2e(c+2)n \log_b n + O(n)$$

and for a random $W = V(p)$,

$$Pr_W[MST(W) \subseteq Q] > 1 - \frac{1}{n^c}.$$

Proof. Let $k = \lceil (c+2) \log_b n \rceil + 1$, $r = \lceil 32ek^2 \ln n \rceil$ and $Q_k^{(p)} = \{(u,v) \in E : \sigma_p(u,v) \geq (1-p)^{k-1}\}$. We will argue that (1) $Q_k^{(p)} \subseteq Q$ with high probability, (2) $Q_k^{(p)}$ is a good covering set, and (3) $|Q| \leq 2ekn + O(n)$.

Let $S_i = V(q)$, $q = 1/k$, and $T_i = MST(S_i)$. As in the proof of Theorem 8, $k \geq 1/p$ (for n large enough), therefore $q \leq p$ and by the boosting lemma, for any $(u,v) \in Q_k^{(p)}$,

$$Pr[(u,v) \in T_i] \geq q^2(1-q)^{k-1} \geq \frac{1}{ek^2}.$$

Denoting by $t(u,v)$ the number of T_i's containing edge (u,v), we get $\mathbf{E}[t(u,v)] \geq r/(ek^2) \geq 32 \ln n$. By Chernoff bound (see [34, Theorem 4.2]; $Pr[X < (1-\delta)\mu] < e^{-\mu\delta^2/2}$), with $\mu \geq 32 \ln n$, $\delta = 1/2$: $Pr[t(u,v) < 16 \ln n] < e^{-4 \ln n} = 1/n^4$, and thus $Pr[\exists (u,v) \in Q_k^{(p)}; t(u,v) < 16 \ln n] < 1/n^2$. Therefore with high probability, all edges in $Q_k^{(p)}$ are included in Q. On the other hand, $Q_k^{(p)}$ contains $MST(W)$ with high probability (with respect to a random $W = V(p)$):

$$Pr[MST(W) \setminus Q_k^{(p)} \neq \emptyset] \leq \sum_{(u,v) \in E \setminus Q_k^{(p)}} Pr[(u,v) \in MST(W)] < n^2 p^2 (1-p)^{k-1} < \frac{1}{n^c}.$$

Now we estimate the size of Q. For $k \geq n/(4e)$, the condition $|Q| \leq 2ekn + O(n)$ is satisifed trivially. So assume $k < n/(4e)$. Since we are sampling $S_i = V(q)$, we have $\mathbf{E}[|S_i|] = qn$, and $\mathbf{E}\left[\sum_{i=1}^r |T_i|\right] \leq \mathbf{E}\left[\sum_{i=1}^r |S_i|\right] \leq rqn$. We can use the Chernoff bound again ([34, Theorem 4.1]; $Pr[X > (1+\delta)\mu] < e^{-\mu\delta^2/3}$), with $\mu \leq rqn$ and $\delta = 10 \ln n/(rq)$:

$$Pr\left[\sum_{i=1}^r |S_i| > (rq + 10 \ln n)n\right] < e^{-100n \ln^2 n/(3rq)} < e^{-n \ln n/ek} < \frac{1}{n^4}.$$

In Q, we include only edges which appear in at least $16 \ln n$ different T_i's, and $|T_i| \leq |S_i|$, so the number of such edges is, with high probability,

$$|Q| \leq \frac{\sum |S_i|}{16 \ln n} \leq \frac{(rq + 10 \ln n)n}{16 \ln n} = 2ekn + O(n).$$

□

3.7 Covering minimum-weight bases in matroids

Next, we consider the variant of the problem where the subgraph is generated by taking a random subset of edges $E(p)$. We approach this problem more generally, in the context of *matroids*. The matroid in this case would be the graphic matroid defined by all forests on the ground set E. In general, consider a weighted matroid $(E, \mathcal{M}, w)$, where $w : E \to \mathbb{R}$. Let m denote the size of the ground set E and n the rank of $\mathcal{M}$, i.e. the size of a largest independent set. If the weights are distinct, then any subset $F \subseteq E$ has a unique minimum-weight basis $MB(F)$, which in the case of graphs corresponds to the minimum-weight spanning forest. These bases satisfy exactly the monotonicity property that we used previously.

Lemma 8. *For an element $e \in E$, let $X = E \setminus \{e\}$ and let $\mathcal{F}$ denote the family of subsets $A \subseteq X$ for which e is in the minimum-weight basis of the matroid induced by $A \cup \{e\}$. Then $\mathcal{F}$ is a down-monotone family:*

$$A \in \mathcal{F}, B \subseteq A \Longrightarrow B \in \mathcal{F}.$$

Proof. If $e \in MB(A \cup \{e\})$, it means that there is no circuit in $A \cup \{e\}$ in which e is the largest-weight element. However, then there is no such circuit in $B \cup \{e\}$ either, and therefore $e \in MB(B \cup \{e\})$. □

Thus, we can apply the same machinery to matroids. Define

$$\sigma_p(e) = Pr_F[e \in MB(F) \mid e \in F]$$

where $F = E(p)$ is a random subset of elements, sampled with probability p. We get statements analogous to the vertex case. It is interesting to notice that the bounds given in these lemmas depend only on the rank of the matroid, irrespective of the size of the ground set.

Lemma 9. *For a weighted matroid $(E, \mathcal{M}, w)$, of rank n, $0 < p < 1$ and $k \geq 1/p$, let*

$$Q_k^{(p)} = \{e \in E : \sigma_p(e) \geq (1-p)^{k-1}\}.$$

Then

$$|Q_k^{(p)}| < ekn.$$

Proof. Sample a random subset $S = E(q)$, each element with probability $q = 1/k \leq p$. For any $e \in Q_k^{(p)}$, $\sigma_p(e) \geq (1-p)^k$, therefore the boosting lemma implies that

$$Pr[e \in MB(S)] \geq q\ \sigma_q(e) \geq \frac{1}{k}\left(1 - \frac{1}{k}\right)^{k-1} > \frac{1}{ek}.$$

Summing over all $e \in Q_k^{(p)}$, we get

$$\mathbf{E}[|MB(S)|] \geq \sum_{e \in Q_k^{(p)}} Pr[e \in MB(S)] > \frac{|Q_k^{(p)}|}{ek}.$$

On the other hand, any independent set in $\mathcal{M}$ has size at most n, therefore $\mathbf{E}[|MB(S)|] \leq n$ which implies $|Q_k^{(p)}| < ekn$. □

Theorem 11. *For any weighted matroid $(E, \mathcal{M}, w)$ of rank n, $0 < p < 1$, $c > 0$, and $b = 1/(1-p)$, there exists a set $Q \subseteq E$ of size*

$$|Q| \leq e(c+1)n \log_b n + O(n/p)$$

such that for a random $F = E(p)$,

$$Pr_F[MB(F) \subseteq Q] > 1 - \frac{1}{n^c}.$$

Proof. Order the elements of E by decreasing values of $\sigma_p(e)$. Partition the sequence into blocks $B_1, B_2, \ldots \subset E$ of size $\lceil en \rceil$. Lemma 9 implies that for any $e \in B_{k+1}$, $k \geq 1/p$:

$$Pr[e \in MB(F)] = p\ \sigma_p(e) < p(1-p)^{k-1}.$$

Take the first $h = \lceil (c+1) \log_b n + 2/p \rceil + 1$ blocks: $Q = \bigcup_{k=1}^{h} B_k$. Then, since $h \geq 1/p$:

$$\begin{aligned} Pr[MB(F) \setminus Q \neq \emptyset] &\leq \sum_{e \in E \setminus Q} Pr[e \in MB(F)] \leq \sum_{k=h+1}^{\infty} \lceil en \rceil p(1-p)^{k-2} \\ &= \lceil en \rceil (1-p)^{h-1} \leq \frac{\lceil en \rceil (1-p)^{2/p}}{n^{c+1}} < \frac{1}{n^c}. \end{aligned}$$

□

The forests in a graph on $n+1$ vertices form a matroid of rank n, and minimum-weight bases correspond to minimum spanning forests. Therefore this solves the edge version of the problem as well:

Corollary 2. *For any weighted graph G on $n+1$ vertices, $0 < p < 1$, $c > 0$ and $b = 1/(1-p)$, there exists a set $Q \subseteq E(G)$ of size*

$$|Q| \leq e(c+1)n \log_b n + O(n/p)$$

such that for a random $F = E(p)$,

$$Pr_F[MST(F) \subseteq Q] > 1 - \frac{1}{n^c}.$$

Also, we get a randomized algorithm finding the covering set for any weighted matroid $(E, \mathcal{M}, w)$; the algorithm makes $O(\log_b n \ln m)$ calls to a minimum-weight basis procedure.

Algorithm 3.

- Let $b = 1/(1-p)$ and $k = \lceil (c+2) \log_b n \rceil + 1$.

- Repeat the following for $i = 1, \ldots, r = \lceil 16ek \ln m \rceil$:
  - Sample $S_i \subseteq E$, each element independently with probability $q = 1/k$.
  - Find $T_i = MB(S_i)$.
- For each edge, include it in Q if it appears in at least $8 \ln m$ different T_i's.

Theorem 12. *Algorithm 3 finds with high probability a set $Q \subseteq E$ such that*

$$|Q| \leq 2e(c+2)n \log_b n + O(n)$$

and for a random $F = E(p)$,

$$Pr_F[MB(F) \subseteq Q] > 1 - \frac{1}{n^c}.$$

Proof. Let $k = \lceil (c+2) \log_b n \rceil + 1$, $r = \lceil 16ek \ln m \rceil$ and $Q_k^{(p)} = \{e \in E : \sigma_p(e) \geq (1-p)^{k-1}\}$. We claim that (1) $Q_k^{(p)} \subseteq Q$ with high probability, (2) $Q_k^{(p)}$ is a good covering set and (3) Q is not too large. As in the proof of Theorem 8, $q = 1/k \leq p$ for n large enough, therefore for any $e \in Q_k^{(p)}$, $S_i = E(q)$ and $T_i = MB(S_i)$, the boosting lemma implies

$$Pr[e \in T_i] = q\ \sigma_q(e) > \frac{1}{ek}.$$

Letting t_e denote the number of T_i's containing element e, we obtain $\mathbf{E}[t_e] \geq r/(ek) \geq 16 \ln m$. By the Chernoff bound, $Pr[t_e < 8 \ln m] < e^{-2 \ln m} = 1/m^2$, implying that $Pr[\exists e \in Q_k^{(p)}; t_e < 8 \ln m] < 1/m$. Therefore with high probability, all edges in $Q_k^{(p)}$ are included in Q.

On the other hand, $Q_k^{(p)}$ contains $MB(F)$ with high probability: Consider the elements in $E \setminus Q$. We order them in a sequence of decreasing values of $\sigma_p(e)$, and again divide them into blocks $B_1, B_2, \ldots$ as before. Since we have included all edges with $\sigma_p(e) \geq (1-p)^{k-1}$, in the first k blocks the values of $\sigma_p(e)$ cannot be larger than as $(1-p)^{k-1}$. However, then the l-th block can have values of $\sigma_p(e)$ at most $\sigma_p(e) \leq (1-p)^{l-2}$ (by Lemma 9). Thus

$$Pr[MB(F) \setminus Q \neq \emptyset] \leq p \left(k(1-p)^{k-1} + \sum_{l=k+1}^{\infty} (1-p)^{l-2} \right) \lceil en \rceil$$

$$\leq \frac{(kp+1)\lceil en \rceil}{n^{c+2}} = \frac{O(\ln n)}{n^{c+1}} < \frac{1}{n^c}$$

for large n. Finally, we estimate the size of Q. We have $\sum_{i=1}^{r} |T_i| \leq rn$. Every element $e \in Q$ appears in $8 \ln m$ different T_i's, therefore

$$|Q| \leq \frac{\sum |T_i|}{8 \ln m} \leq 2ekn + O(n).$$

□

3.8 Lower bounds

For both variants of the problem, we have a closely matching lower bound on the size of Q, even if we want to achieve a constant probability of covering the MST. We get a lower bound of $\Omega(n \log_b(n/\ln n))$ for $p > \ln n/n$ in the edge case and $\Omega(n \log_b(pn/5))$ for $p > 5/n$ in the vertex case. Both bounds reduce to $\Omega(n \log_b n)$, for a wide range of p, namely the lower bound of $\Omega(n \log_b n)$ holds for $p \geq 1/n^\gamma, \gamma < 1$.

The constructions for the vertex and edge variants are different; first let's describe the edge variant which is simpler.

Lemma 10. *For any $n > e^{e^2}$, $\frac{\ln n}{n} \leq p < 1$, there is a weighted graph G on n vertices, such that if*

$$Pr[MST(E(p)) \subseteq Q] \geq \frac{1}{e}$$

then $|Q| > ln - \binom{l+1}{2}$ where $l = \lfloor \log_b(n/\ln n) \rfloor$.

Proof. Consider the complete graph K_n with edge weights ordered lexicographically. For $i < j$, let

$$w_{ij} = ni + j$$

(see Figure 2-1). Let $F = E(p)$ be a random subset of edges. For each edge (j,k), $j < k$, consider an event A_{jk}, which occurs when

$$(j,k) \in F \ \& \ \forall i < j; (i,k) \notin F.$$

Due to the ordering of edge weights, A_{jk} implies that $(j,k) \in MST(F)$, since it is the lightest edge in F, incident with vertex k. Also, for given k, A_{jk} can occur only for one value of j. For a set Q of given size, we estimate the probability that A_{jk} occurs for some $(j,k) \in E \setminus Q$:

Let $J_k = \{j : j < k, (j,k) \in E \setminus Q\}$. Since the events A_{jk} for different elements of J_k are disjoint,

$$Pr[\bigcup_{j \in J_k} A_{jk}] = \sum_{j \in J_k} p(1-p)^{j-1}.$$

The events $\bigcup_{j \in J_k} A_{jk}$ for different k's are mutually independent, since the set of edges involved for different J_k's are disjoint. Therefore:

$$Pr[MST(F) \subseteq Q] \leq Pr[\bigcap_{(j,k) \in E \setminus Q} \overline{A_{jk}}] = \prod_k Pr[\bigcap_{j \in J_k} \overline{A_{jk}}]$$

$$= \prod_k (1 - \sum_{j \in J_k} p(1-p)^{j-1}) \leq \exp\left(- \sum_{(j,k) \in E \setminus Q} p(1-p)^{j-1} \right).$$

For a given size of Q, the last expression is maximized when Q contains edges (j,k) with minimum possible values of j. Assume that Q contains all the edges (j,k)

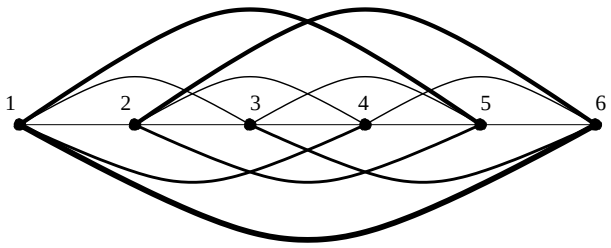

Figure 3-2: The lower bound example for random sampling of vertices. Edge weights are marked by thickness.

for $j = 1, 2, \ldots l$. Then $|Q| = \sum_{j=1}^{l} (n-j) = ln - \binom{l+1}{2}$ and

$$\sum_{(j,k)\in E\setminus Q} p(1-p)^{j-1} = \sum_{j=l+1}^{n-1} (n-j)p(1-p)^{j-1}.$$

Let's denote this sum by $S(l)$. As can be verified by backward induction on l,

$$S(l) = \left(n - l - \frac{1}{p}\right)(1-p)^l + \frac{1}{p}(1-p)^n.$$

We have that for any Q of size at most $ln - \binom{l+1}{2}$, $Pr[MST(F) \subseteq Q] \leq e^{-S(l)}$.

Let's choose $l = \lfloor \log_b(n/\ln n) \rfloor$. Then, for $p \geq \frac{\ln n}{n}$,

$$S(l) \geq \left(n - \log_b\left(\frac{n}{\ln n}\right) - \frac{1}{p}\right)\frac{\ln n}{n} \geq \left(n - \frac{\ln n - \ln\ln n + 1}{p}\right)\frac{\ln n}{n} \geq \ln\ln n - 1.$$

Therefore, for any set Q of size at most $ln - \binom{l+1}{2}$, $Pr[MST(W) \subseteq Q] \leq e^{-S(l)} < 1/e$ for $n > e^{e^2}$.

□

Lemma 11. *For any $n > 5$, and $p \geq 5/n$, there exists a weighted graph G on n vertices, such that if*

$$Pr[MST(V(p)) \subseteq Q] \geq \frac{1}{e}$$

then $|Q| > ln - \binom{l+1}{2}$ where $l = \lfloor \log_b(np/5) \rfloor$.

Proof. Let G be the complete graph K_n. Consider the vertices placed on a line uniformly and define edge weights by distances between pairs of vertices, breaking ties arbitrarily, for example:

$$w_{ij} = n|j-i| + i.$$

Let $W = V(p)$ be a random subset of vertices. For each edge (j,k), $j < k$, consider an event A_{jk} as a boolean function of W:

$$A_{jk}(W) \iff j \in W,\ k \in W\ \&\ \forall i; j < i < k \Rightarrow i \notin W.$$

This event is equivalent to $(j,k) \in MST(W)$, since (j,k) must be in $G[W]$ and no path connecting j,k via a vertex in between can be in $G[W]$. However, we have to be more careful here, because these events are not necessarily independent, which might ruin our bound on the probability of their union. Therefore, we have to impose an additional condition which we deal with later. Assume that C is a set of edges satisfying

(*) *There is no pair of edges* $(i,j),(j,k) \in C$ *such that* $i < j < k$.

Then we claim that the events $\overline{A_{jk}}$ for $(j,k) \in C$ are never positively correlated. More specifically, if $(u_0,v_0),(u_1,v_1),\dots(u_k,v_k) \in C, u_i < v_i$, then

$$Pr[\overline{A_{u_1v_1}} \cap \overline{A_{u_2v_2}} \cap \dots \overline{A_{u_kv_k}} \mid \overline{A_{u_0v_0}}] \leq Pr[\overline{A_{u_1v_1}} \cap \overline{A_{u_2v_2}} \cap \dots \overline{A_{u_kv_k}}]. \tag{3.3}$$

We prove this in the following way: For any $W \subseteq V$, define $W' = W \cup \{u_0,v_0\} \setminus \{i : u_0 < i < v_0\}$. Then $A_{u_0v_0}(W')$ is true by definition. In fact, if $W = V(p)$ is a random subset, W' is a random subset sampled in the same way as W, but conditioned on $A_{u_0v_0}$.

Now consider the other edges, $(u_1,v_1),\dots(u_k,v_k)$. Let's call an edge (u_i,v_i) "interfering" with (u_0,v_0), if its interval $[u_i,v_i]$ intersects $[u_0,v_0]$. Property $(*)$ implies that the intervals $[u_0,v_0]$, $[u_i,v_i]$ cannot share exactly one point, so either one of u_i,v_i is an internal point of $[u_0,v_0]$, or one of u_0,v_0 is an internal point of $[u_i,v_i]$. Either way, $A_{u_iv_i}(W')$ cannot be true, because then u_i and v_i ought to be in W' and all vertices inside $[u_i,v_i]$ ought not to be in W' which is impossible. Therefore, $A_{u_iv_i}(W')$ is always false for (u_i,v_i) interfering with (u_0,v_0). On the other hand, if an edge (u_i,v_i) is not interfering with (u_0,v_0), then $A_{u_iv_i}(W')$ if and only if $A_{u_iv_i}(W)$, because W' does not differ from W on the vertices outside of $[u_0,v_0]$.

Thus we have demonstrated that in any case, $\overline{A_{u_iv_i}(W)} \Rightarrow \overline{A_{u_iv_i}(W')}$, and W' corresponds to random sampling conditioned on $A_{u_0v_0}$, which implies

$$Pr[\overline{A_{u_1v_1}} \cap \overline{A_{u_2v_2}} \cap \dots \overline{A_{u_kv_k}} \mid A_{u_0v_0}] \geq Pr[\overline{A_{u_1v_1}} \cap \overline{A_{u_2v_2}} \cap \dots \overline{A_{u_kv_k}}].$$

This is equivalent to Eq. 3.3. As a consequence, we get

$$Pr[\bigcap_{e \in C} \overline{A_e}] \leq \prod_{e \in C} Pr[\overline{A_e}].$$

For a set C satisfying $(*)$, we can now estimate the probability that none of these edges appear in $MST(W)$:

$$Pr[C \cap MST(W) = \emptyset] = Pr[\bigcap_{(j,k) \in C} \overline{A_{jk}}] \leq \prod_{(j,k) \in C} Pr[\overline{A_{jk}}]$$

$$= \prod_{(j,k) \in C} (1 - p^2(1-p)^{|k-j|-1}) < \exp\left(-\sum_{(j,k) \in C} p^2(1-p)^{|k-j|-1}\right).$$

Suppose that Q has size at most $\sum_{j=1}^{l}(n-j) = ln - \binom{l+1}{2}$. The optimal way to

minimize $\sum_{(j,k)\in E\setminus Q} p^2(1-p)^{|k-j|-1}$ is to choose Q to contain all the edges of length at most l. Then we have

$$\sum_{(j,k)\in E\setminus Q} p^2(1-p)^{|k-j|-1} = \sum_{j=l+1}^{n-1} (n-j)p^2(1-p)^{j-1} = p\ S(l)$$

where $S(l)$ is defined in the previous proof, $S(l) = (n-l-\frac{1}{p})(1-p)^l + \frac{1}{p}(1-p)^n$. We choose $l = \lfloor \log_b(np/5) \rfloor$ and then:

$$p\,S(l) \geq (p(n-l)-1)(1-p)^l \geq (p(n-\log_b(np/5))-1)\frac{5}{np} \geq 5\left(1 - \frac{\ln(np/5)+1}{np}\right) \geq 4.$$

Thus we have $\sum_{(j,k)\in E\setminus Q} p^2(1-p)^{|k-j|-1} \geq 4$ for any Q of size at most $ln - \binom{l+1}{2}$. Now we apply the probabilistic method to choose a suitable subset of $E \setminus Q$. We sample a uniformly random subset of vertices S. Let $C = \{(j,k) \in E\setminus Q : j < k, j \in S, k \notin S\}$. For each edge in $E \setminus Q$, there is probability $1/4$ that it appears in C. Therefore

$$\mathbf{E}[\sum_{(j,k)\in C} p^2(1-p)^{|k-j|-1}] \geq \frac{1}{4} \sum_{(j,k)\in E\setminus Q} p^2(1-p)^{|k-j|-1} \geq 1$$

and there exists a set C which achieves at least this expectation. Due to the construction of C, it satisfies condition (*), and we can conclude that

$$Pr[MST(W) \subseteq Q] \leq Pr[C \cap MST(W) = \emptyset] < e^{-1}.$$

This is a contradiction, and so Q must be larger than $ln - \binom{l+1}{2}$. □

Chapter 4

Metric approximation for random subgraphs

4.1 Covering shortest paths

We continue in the spirit of Chapter 3 where the goal was to find a sparse set of edges covering with high probability the solution of an optimization problem (MST, in particular) for a random subgraph. Consider another fundamental optimization problem on graphs: the Shortest Path.

Assume that we have a graph G and two designated vertices $s, t \in V$. Recall that we denote by $V(p)$ a random subset of V where each vertex is sampled independently with probability p. We would like to find a set of edges Q such that for a random induced subgraph $G[W], W = V(p) \cup \{s, t\}$, Q contains a shortest $s-t$ path in $G[W]$ with high probability. It turns out that in contrast to the MST problem, Shortest Path is not amenable to good covering sets.

Example. Consider a graph $G = (V, E)$ where

- $V = \bigcup_{i=1}^{k} A_i \cup \bigcup_{i=1}^{k} B_i$
- $E = \bigcup_{i=1}^{k} E(A_i) \cup \bigcup_{i=1}^{k} E(B_i) \cup H$.
- $A_i = \{s, a_{i1}, a_{i2}, \ldots, a_{il}\}$
- $E(A_i)$ is a path $s - a_{i1} - a_{i2} - \ldots - a_{il}$
- $B_i = \{t, b_{i1}, b_{i2}, \ldots, b_{il}\}$
- $E(B_i)$ is a path $t - b_{i1} - b_{i2} - \ldots - b_{il}$
- H is a complete bipartite graph $\{(a_{il}, b_{jl}) : 1 \leq i, j \leq k\}$.
- The edges in $E(A_i)$ and $E(B_i)$ have zero length, while the edges in H have unit length.

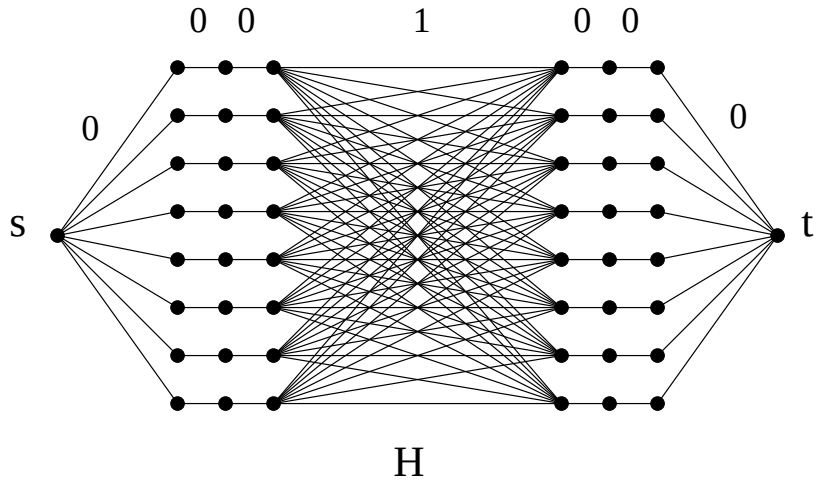

Figure 4-1: A counterexample to $s-t$-shortest path covering, for $k = 8$ and $l = 3$.

We set $k = 2^l, n = |V| = l2^{l+1} + 2$. Sample $W \subseteq V$, each vertex with probability $1/2$ except s, t which are always in W. Consider the event that A_i is the unique path among $A_1, A_2, \ldots, A_k$ which survives in $G[W]$: this happens with probability $2^{-l}(1 - 2^{-l})^{k-1} > 1/ek$. Independently, B_j is the unique surviving path among $B_1, B_2, \ldots, B_k$, with probability $1/ek$. Thus with probability $1/(ek)^2$, there is a unique shortest $s-t$ path (of length 1), using the surviving paths A_i, B_j and the edge $(a_{il}, b_{jl}) \in H$. This holds for every edge in H with the same probability and the corresponding events are disjoint. If $h = |H \setminus Q|$, there is probability at least $h/(ek)^2$ that Q doesn't contain the shortest path. Therefore Q must contain $\Omega(k^2)$ edges in H, for instance $k^2/2$, otherwise the probability of missing the shortest path is at least $1/2e^2$. We conclude:

Lemma 12. *There are graphs on n vertices such that if Q covers with probability at least $1/2e^2$ the shortest $s-t$ path in $G[W]$, $W = V(1/2) \cup \{s, t\}$, then*

$$|Q| = \Omega\left(\frac{n^2}{\log^2 n}\right).$$

Moreover, note that if Q does not contain the shortest path, the next shortest connection between s and t has length at least 3. So in fact, we need $\Omega(n^2/\log^2 n)$ edges even if we want Q to contain some *approximately-shortest* $s-t$ path, for any stretch factor $c < 3$.

It is natural to ask whether it is possible to find a sparse set of edges Q which contains at least an approximately-shortest $s-t$ path with high probability, for some stretch factor $c \geq 3$. In the rest of this chapter, we address this problem in more generality.

4.2 Approximating the shortest-path metric

Consider the shortest-path problem more generally: Is it possible to find a sparse set of edges Q such that for a random subgraph $G[W]$, the shortest-path metric induced by Q in $G[W]$ approximates the shortest-path metric of $G[W]$ itself? This would preserve approximate solutions of not only the shortest $s-t$-path problem, but also any graph optimization problem based on the shortest-path metric (such as Steiner Tree or Traveling Salesman).

Definition 1. *A set of edges Q is c-metric-approximating for a certain distribution of random subgraphs $H \subset G$, if with high probability, for any $u-v$ path in H there is a $u-v$ path in H, using only edges of Q, of length expanded at most by a factor of c.*

Note that this resembles the definition of a *c-spanner* - a subgraph approximating the metric of the original graph to within a constant factor c. However, our requirements are stronger - Q should be a c-spanner even after some vertices (or edges) have been randomly destroyed, with high probability.

The known lower bounds on spanners apply to our case as well.

Lemma 13. *There are graphs G for any $n = |V(G)|$ and $c \geq 1$ such that any c-metric-approximating set Q for $H = G[W]$, $W = V(p)$, $p > 0$ constant, has size*

- $|Q| = \Omega(n^2)$ *for* $c < 3$.
- $|Q| = \Omega(n^{3/2})$ *for* $c < 5$.
- $|Q| = \Omega(n^{1+1/(g-2)})$ *for* $c < g-1$, $g \geq 8$ *even.*

Proof. Consider a graph G without cycles of length $\leq c+1$ (i.e., the *girth* of G, the length of its shortest cycle, is $g(G) \geq c+2$). It is known [3] that for $g(G) = 4$, G can have $\Omega(n^2)$ edges (the complete bipartite graph), for $g(G) = 6$, G can have $\Omega(n^{3/2})$ edges (the incidence graph of a finite projective plane), and for $g(G) \geq 8$ even, G can have $\Omega(n^{1+1/(g-2)})$ edges [35]. Then Q must contain *all* edges of G. Otherwise, there is an edge $(u,v) \in E \setminus Q$, which appears in $G[W]$ with constant probability, and the shortest possible $u-v$ connection in Q has length $g-1$. □

4.3 Construction of metric-approximating sets for random vertex-induced subgraphs

Now let's turn to the construction of a good metric-approximating set. The question is related to MST covering in the sense that we must ensure that any edge in $E \setminus Q$ is covered by another path in Q, with high probability. But here the requirement is stronger - rather than covering by any path consisting of edges of smaller weight, we care about the total length of the covering path.

There is one construction we mentioned already that gives a non-trivial result. Recall Algorithm 1 described in Section 3.1. As Lemma 1 claims, this algorithm

finds an edge set Q of size $O(n^{3/2} \log n)$ which covers the MST of a random vertex-induced subgraph with high probability. Moreover, the proof implies that with high probability, every edge $(u, v) \in E(G[W]) \setminus Q$ is covered by a path in $E(G[W]) \cap Q$ *containing at most 3 edges.* Since these edges also have length at most that of (u, v), this path also provides a 3-approximation in the metric sense. Thus we obtain immediately:

Lemma 14. *For any graph G with edge lengths, a 3-metric-approximating set $Q \subseteq E$ (for uniformly random vertex-induced subgraphs) of size $O(n^{3/2} \log n)$ can be found in polynomial time.*

This result for $c = 3$ is best possible up to a logarithmic factor, considering Lemma 13. The remaining question is whether we can find sparser c-metric approximating sets for larger values of c. Also, consider random subgraphs $G[W], W = V(p)$ for an arbitrary (although constant) sampling probability p.

The essential obstacle to applying Algorithm 1 with longer covering paths is that these paths need not be vertex-disjoint. For instance, there can be arbitrarily many $u - v$ paths of length 4 with a vertex-cut of size 1. This would destroy our objective to guarantee a high probability of survival for at least one path.

The solution is to run the same algorithm repeatedly on random induced subgraphs. This will force the covering paths to be vertex-disjoint with a certain probability and a careful tuning of the sampling parameters will yield a good bound on the size of a c-metric approximating set.

Algorithm 4. (given parameters $t, c \in \mathbb{Z}_+, q \in (0, 1)$)

- Repeat the following steps in stages $i = 1, 2, \ldots, t$.
- Set $Q_i = \emptyset$ and sample a random subset of vertices $S_i = V(q)$.
- Process the edges of $G[S_i]$ in the order of increasing edge lengths.
- Include each edge in Q_i, unless it is covered by a path of at most c edges in Q_i.
- Set $Q = \bigcup_{i=1}^{t} Q_i$.

Our goal here is to cover each edge in $E \setminus Q$ by $\Omega(p^{-c} \log n)$ vertex-disjoint paths of length stretched by at most c. For $1 \leq j \leq t$, consider an edge $\{u, v\} \in E \setminus \bigcup_{i=1}^{j} Q_i$ and suppose that in $\bigcup_{i=1}^{j} Q_i$ it has been covered by $k < 8p^{-c} \log n$ vertex-disjoint paths with at most c edges. Define a set of vertices $R_j(u, v)$ which contains the internal vertices of these k paths, does not contain u and v, and contains some additional vertices (for example, the lexicographically smallest) so that the cardinality of $R_j(u, v)$ is always $r = \lfloor 8cp^{-c} \log n \rfloor$. If $\{u, v\}$ has been already covered by $8p^{-c} \log n$ vertex-disjoint paths, define $R_j(u, v)$ of size r arbitrarily.

Call S_{j+1} "good for $\{u, v\}$", if $u, v \in S_{j+1}$ and $R_j(u, v) \cap S_{j+1} = \emptyset$.

$$Pr[S_{j+1} \text{ is good for } \{\text{u,v}\}] = q^2(1 - q)^r.$$

We choose $q = 1/r$, assuming $r \geq 2$, which implies

$$Pr[S_{j+1} \text{ is good for } \{\text{u,v}\}] \geq \frac{1}{4r^2}.$$

Observe that if S_{j+1} is good for $\{u, v\}$ then either we include $\{u, v\}$ in Q, or we cover it by a new vertex-disjoint path, unless it is covered by $8p^{-c} \log n$ paths already. We have a total of t stages. We estimate the number of good stages for $\{u, v\}$:

$$\mu = \mathbf{E}[\#\text{good stages for } \{\text{u,v}\}] \geq \frac{t}{4r^2}.$$

We choose $t = \lceil 64r^2\ p^{-c} \log n \rceil$ so that $\mu \geq 16p^{-c} \log n$. The events of "a stage being good for {u,v}" are independent for different stages. By the Chernoff inequality,

$$Pr[\#\text{stages good for } \{\text{u,v}\} < \mu/2] < e^{-\mu/8} \leq n^{-2p^{-c}}$$

$$Pr[\exists \{u, v\} \in E;\ \#\text{good stages for } \{\text{u,v}\} < \mu/2] < n^{2-2p^{-c}} << 1.$$

So there are at least $\mu/2 \geq 8p^{-c} \log n$ good stages for every edge, with high probability with respect to our sampling of $S_1, \ldots, S_t$. Assuming that this happens, since every good stage either includes $\{u, v\} \in Q$ or finds a new vertex-disjoint path covering $\{u, v\}$, in the end every edge in $E \setminus Q$ has at least $8p^{-c} \log n$ vertex-disjoint covering paths each consisting of at most c edges. Each path survives in $G[W]$, where $W = V(p)$, with probability at least p^c. This implies

$$Pr_W[Q \cap E(G[W]) \text{ is a } c\text{-spanner for } G[W]]$$

$$\geq 1 - \sum_{e \in E \setminus Q} (1 - p^c)^{8p^{-c} \log n} > 1 - \sum_{e \in E \setminus Q} \frac{1}{n^8} > 1 - \frac{1}{n^6}.$$

Again, the size of Q is constrained by the absence of short cycles. Each Q_i is a set of edges of girth $g(Q_i) \geq c + 2$. Moreover, it is a subgraph on the vertices of S_i where

$$\mathbf{E}[|S_i|] = qn = \frac{n}{r}$$

and by the Chernoff inequality

$$Pr[|S_i| > 2n/r] < e^{-n/3r},$$

$$Pr[\exists i \leq t; |S_i| > 2n/r] < t\ e^{-n/3r}$$

which will be very small for our choice of parameters ($r, t = polylog(n)$). Thus with high probability, we also have $|S_i| \leq 2n/r$. The size of a set of edges on S_i of girth $g(Q_i) \geq c + 2$ can be at most

$$|Q_i| \leq |S_i|(1 + |S_i|^{2/c}) \leq \frac{2n}{r}(1 + n^{2/c})$$

(see [3]). Thus the total number of edges selected by the algorithm is:

$$|Q| \leq \frac{2nt}{r}(1+n^{2/c}) \leq (2^9+o(1))\ c\ p^{-2c}\ n^{1+2/c}\log^2 n$$

with high probability. We conclude:

Theorem 13. *For any graph G with edge lengths, Algorithm 4 with integer $c \geq 3$ and $t = \lceil 2^{12}c^2\ p^{-3c}\log^3 n\rceil$ finds with high probability a c-metric-approximating set $Q \subseteq E$ (for random subgraphs induced by $W = V(p)$) such that*

$$|Q| = O\left(p^{-2c}n^{1+2/c}\log^2 n\right).$$

It should be noted that for p constant, the gap between our upper and lower bounds is $O(n^{1/c}\log^2 n)$. However, the factor of $n^{1/c}$ arises from the lack of tight bounds on graphs of given girth rather than poor performance of our algorithm. Alternatively, we could formulate our bounds in terms of the maximal number of edges for graphs of girth g (which is not known precisely). Denoting this number by $m(n,g)$, we would obtain that our c-metric-approximating set has size $O(m(n,c+2)\log^2 n)$, while the lower bound is exactly $m(n,c+2)$. This is also the standard lower bound for spanners. Thus the price we pay for having a "robust spanner", resistant to random failures of vertices, is $O(\log^2 n)$.

4.4 Construction of metric-approximating sets for random edge-induced subgraphs

The solution is more straightforward for random edge-induced subgraphs. We invoke Algorithm 4 once again.

Theorem 14. *For any graph G with edge lengths, Algorithm 4 with $r = \lceil 4p^{-c}\ln n\rceil$ and integer $c \geq 3$ finds with high probability a c-metric-approximating set $Q \subseteq E$ (for random subgraphs induced by $F = E(p)$) such that*

$$|Q| = O\left(p^{-c}n^{1+2/c}\log n\right).$$

Proof. Recall the proof of Lemma 2. The bound on $|Q|$ follows again from the girth of Q_i in each stage which is at least $c+2$. Similarly, every edge $e \in E\setminus Q$ is covered by r *edge-disjoint* paths whose length approximates that of e by a factor of at most c. Each of them has probability of survival at least p^c. Here, these events are independent because of edge-disjointness. Consequently, the probability that none of these paths survive in F is at most $(1-p^c)^r < 1/n^4$. By the union bound, all edges in $F\setminus Q$ have a c-approximate metric covering in $Q\cap F$ with probability at least $1-1/n^2$. □

Chapter 5

Stochastic Optimization and Adaptivity

Let us introduce a class of optimization problems with randomness on the input. Our basic setting is that the input consists of a collection $\mathcal{I}$ of atomic *items* which are characterized by certain (possibly random) properties. We call an item *instantiated* if its properties have been specified by drawing a sample from the respective probability distribution. The random properties for different items are assumed to be mutually independent. A collections of items with instantiated properties can be *feasible* or *infeasible* - this depends on the properties of included items, for example their total size. We seek a strategy (or *policy*) that selects a feasible set of items of optimum cost.

The stochastic aspect of our optimization problems arises from the randomness of item properties. For a given set $\mathcal{A} \subseteq \mathcal{I}$, it is not a priori clear whether it is feasible or infeasible. Our basic assumption is that a policy has some information on the probability distribution of item properties, but their exact instantiation is not known beforehand. However, once an item is included in the solution, its properties are instantiated and made known to the policy. A policy can then use this knowledge to adapt its decisions in the future. Hence, we distinguish two kinds of policies: *adaptive* and *non-adaptive*. An adaptive policy makes its decisions based on the properties of items selected so far, while a non-adaptive policy must specify a fixed sequence of items in advances, regardless of their instantiated sizes.

Note. There is no reference to computational efficiency in our notion of a policy. An adaptive policy may be viewed as an oracle specifying which item should be inserted for a given configuration. The existence of a policy does not necessarily mean that it can be realized algorithmically. Algorithmic efficiency of finding an optimal or approximately optimal policy (in the usual sense of polynomial-time computability) is another aspect of a stochastic optimization problem and it will be addressed here as well.

5.1 Stochastic Knapsack

Let's turn to a more specific problem. The starting point of our investigation is the Stochastic Knapsack problem, introduced by Brian Dean. The classical knapsack problem involves n items with values $v_1 \dots v_n \in \mathbb{R}_+$ and sizes $s_1 \dots s_n \in \mathbb{R}_+$, and asks for a maximum-value set of items that fits within given capacity (each item can be used at most once). In our stochastic generalization, we consider deterministic values and independently random sizes.

Definition 2 (Stochastic Knapsack)**.** *An instance of Stochastic Knapsack is given by a collection of items with deterministic values $v_1, v_2, v_3, \dots \in \mathbb{R}_+$ and random sizes $s_1, s_2, s_3, \dots$ which are independent random variables with values in $\mathbb{R}_+$. The actual size of an item is unknown until we instantiate it by attempting to place it in the knapsack. The knapsack capacity is assumed to be normalized to* 1*.*

A non-adaptive policy is an ordering of items to insert. An adaptive policy is a function $\mathcal{P} : 2^{[n]} \times \mathbb{R}_+ \to [n]$. The interpretation of $\mathcal{P}$ is that given a configuration $(\mathcal{J}, c)$ where $\mathcal{J}$ represents the items still available and c the remaining capacity, $\mathcal{P}(\mathcal{J}, c)$ determines which item should be chosen next among the items in $\mathcal{J}$.

A policy inserts items until the knapsack capacity is exceeded. For each item inserted within the capacity, the policy receives a profit equal to the value of the inserted item. Our objective function is the expected value collected by a policy.

This problem is motivated by the application of scheduling a maximum-value subset of n tasks with uncertain durations within a fixed amount of time. The conditions of our stochastic setting arise from the following assumptions: (1) jobs have random running times, (2) once a job has been scheduled, we cannot remove it, (3) once a job has been completed, its running time is known exactly. A scheduling policy can have the flexibility of choosing the next job based on the time remaining before the deadline, and the objective here is to maximize the profit received for jobs processed before a given deadline.

To be more precise, let's specify what information a policy has on the probability distribution of job/item sizes; in this thesis, we restrict ourselves to the following assumption:

For each item, a policy knows beforehand the probability that the item alone fits within the capacity, and the expectation of the item's size truncated to the capacity,

$$\mu_i = \mathbf{E}[\min\{s_i, 1\}].$$

The reason for this assumption will be apparent later. Intuitively, knowing the actual expected size $\mathbf{E}[s_i]$ is not very useful since beyond the knapsack capacity, the size of an item does not make any difference.

Let's define a notation for *Bernoulli random variables.*

Definition 3. *$Be(p)$ denotes a random variable which attains value* 1 *with probability p and value* 0 *with probability $1 - p$.*

Example. Two items of size 20 $Be(0.1)$ and 1+10 $Be(0.1)$ have the same expected size (2) and even the same probability of fitting (each alone fits with probability 0.9). Yet their behavior is very different: assuming that their values are equal, the first item is certainly more profitable for our purposes. Note that the *mean truncated size* is $\mu_1 = 0.1$ for the first item and $\mu_2 = 1$ for the second item.

Given such a Stochastic Knapsack instance, what are the questions we are interested in?

1. What is the optimum adaptive policy? (I.e., the adaptive policy maximizing the expected value achieved for items successfully inserted.) Can we characterize the optimum adaptive policy?

2. What is a good non-adaptive policy? (I.e., a good ordering of items to be inserted sequentially.)

3. What is the largest possible ratio of expected values achieved by optimal adaptive / non-adaptive policies? We call this factor the *adaptivity gap*. (Note that this is independent of any computational assumptions; it refers merely to the existence of policies, which may not be computable efficiently.)

4. What is the best adaptive or non-adaptive policy that can be found in polynomial time? Both positive (approximation) and negative (hardness) results are interesting.

In [15], we obtained the following results:

- There are instances where an adaptive policy can perform better than any non-adaptive policy, by a constant factor. There are examples where the adaptivity gap is equal to 5/4.

- There is an efficient greedy algorithm that computes a non-adaptive policy with expected value is at least 1/7 that of an optimal adaptive policy, thereby giving an upper bound of 7 on the adaptivity gap for Stochastic Knapsack.

- There is a polynomial-time adaptive policy (i.e., a policy that takes polynomial time to decide about the next item to insert in the knapsack given the sizes of items inserted so far) whose expected value is at least $(1/5 - \epsilon)$ times that of an optimal adaptive policy, for any constant $\epsilon > 0$. However, this policy requires additional information on the probability distribution of item sizes. In principle, having more information may improve approximation factors, but we shall not investigate this issue here.

- Regarding the computational complexity of Stochastic Knapsack, it turns out that many natural questions are PSPACE-hard. Namely, it is PSPACE-hard to decide whether there is an optimal adaptive policy which fills the knapsack always exactly to its capacity.

In this thesis, we address the Stochastic Knapsack in Chapter 6, before we move on to more general classes of stochastic optimization problems. We show an improved analysis of the adaptivity gap which yields a constant factor of 4. In Chapter 9, we present the relation of stochastic optimization to PSPACE, and the complexity of Stochastic Knapsack is also discussed there. For more information on Stochastic Knapsack, the reader can also consult Brian Dean's PhD thesis [14].

5.2 Stochastic Packing and Covering problems

Now we define a more general class of problems we are interested in. They can be categorized as *Stochastic Packing* and *Stochastic Covering*. In both cases, the input comes in the form of a collection of *items*. Item i has a scalar value v_i and a vector size $\vec{S}(i)$. Unless otherwise noted, we assume that $\vec{S}(i)$ is a random vector with nonnegative components, while v_i is a deterministic nonnegative number. The random size vectors of different items are assumed independent. Occasionally, we will consider random values as well, possibly correlated with the respective item size, but the pairs $(v_i, \vec{S}(i))$ as random vectors are *always* mutually independent.

We start with the deterministic form of a general packing problem, which is known under the name of a *Packing Integer Program* (see [40], [10]).

Definition 4 (PIP)**.** *Given a matrix $A \in \mathbb{R}_+^{d\times n}$ and vectors $\vec{b} \in \mathbb{R}_+^d, \vec{v} \in \mathbb{R}_+^n$, a Packing Integer Program (PIP) is the problem of maximizing $\vec{v} \cdot \vec{x}$ subject to $A\vec{x} \leq \vec{b}$ and $\vec{x} \in \{0,1\}^n$.*

This is a combinatorial problem in a very general form, encapsulating many combinatorial problems such as Hypergraph Matching (a.k.a. Set Packing), b-matching, disjoint paths in graphs, Maximum Clique and Maximum Independent Set. In combinatorial optimization, it is often useful to consider a *linear relaxation* of a combinatorial problem, where we allow fractional parts of items to be used in a solution. This means simply relaxing the condition $\vec{x} \in \{0,1\}^n$ to $\vec{x} \in [0,1]^n$. The optimum of the relaxed problem can be greater than that of the original problem. The ratio of the two optima is called the *integrality gap*.

Definition 5. *For an instance of PIP, define*

- $OPT = \max\left\{\vec{v} \cdot \vec{x} :\ A\vec{x} \leq \vec{b},\ \vec{x} \in \{0,1\}^n\right\}$, *the optimum of the packing integer program.*
- $LP = \max\left\{\vec{v} \cdot \vec{x} :\ A\vec{x} \leq \vec{b},\ \vec{x} \in [0,1]^n\right\}$, *the optimum value of the respective linear relaxation.*
- *Then the integrality gap is* $\omega = LP/OPT$.

For a class of PIP instances, define ω^* as the supremum of possible integrality gaps.

In analogy to Stochastic Knapsack, we consider a stochastic generalization of PIP where items have random vector sizes. The special variants would correspond to Set Packing with random sets, b-matching on a random hypergraph, etc.

Definition 6 (Stochastic Packing)**.** *Stochastic Packing (SP) is a stochastic variant of a PIP where A is a random matrix whose columns are independent random vectors, denoted $\vec{S}(1), \ldots \vec{S}(n)$. The capacity vector $\vec{b}$ is considered deterministic. We view the columns as items to be selected for a feasible solution. A feasible solution is a set of items F such that $\sum_{i \in F} \vec{S}(i) \leq \vec{b}$. The value of $\vec{S}(i)$ is instantiated and fixed once we include item i in F. Once this decision is made, the item cannot be removed. Whenever the condition $\sum_{i \in F} \vec{S}(i) \leq \vec{b}$ is violated, no further items can be inserted, and no value is received for the overflowing item.*

We consider 4 classes of Stochastic Packing problems: (1) General Stochastic Packing where no restrictions are placed on item sizes or capacity; by scaling, we can assume that $\vec{b} = (1, 1, \ldots, 1)$. (2) Restricted Stochastic Packing where the $\vec{S}(i)$ have values in $[0,1]^d$ and $\vec{b} \in \mathbb{R}^d, b_j \geq 1$. (3) Stochastic Set Packing where the $\vec{S}(i)$ have values in $\{0,1\}^d$ and $\vec{b} = (1, 1, \ldots, 1)$. (4) Stochastic b-matching where the $\vec{S}(i)$ have values in $\{0,1\}^d$ and $\vec{b} \in \mathbb{Z}^d, b_j \geq 1$.

As a dual to Stochastic Packing, we define Stochastic Covering as a generalization of Set Cover and Covering Integer Programs.

Definition 7 (CIP)**.** *Given a collection of subsets $\mathcal{F} = \{S_1, S_2, \ldots, S_s\}, \subseteq \mathcal{P}(S)$, Set Cover is the problem of selecting as few as possible so that their union is equal to S.*

More generally, given a matrix $A \in \mathbb{R}_+^{n \times d}$ and vectors $\vec{b} \in \mathbb{R}_+^d, \vec{v} \in \mathbb{R}_+^n$, a Covering Integer Program (CIP) is the problem of minimizing $\vec{v} \cdot \vec{x}$ subject to $A\vec{x} \geq \vec{b}$ and $\vec{x} \in \{0,1\}^d$.

Definition 8. *For an instance of CIP, define*

- $OPT = \min\left\{\vec{v} \cdot \vec{x} : \ A\vec{x} \geq \vec{b}, \ \vec{x} \in \{0,1\}^n\right\}$, *the optimum of the packing integer program.*
- $LP = \min\left\{\vec{v} \cdot \vec{x} : \ A\vec{x} \geq \vec{b}, \ \vec{x} \in [0,1]^n\right\}$, *the optimum value of the respective linear relaxation.*
- *Then the integrality gap is: $\omega = OPT/LP$.*

For a class of CIP instances, define ω^* as the supremum of possible integrality gaps.

Definition 9 (Stochastic Covering)**.** *Stochastic Covering (SP) is a stochastic variant of a CIP where A is a random matrix whose columns are independent random vectors, denoted $\vec{S}(1), \ldots \vec{S}(n)$. The capacity vector $\vec{b}$ is considered deterministic. A feasible solution is a set of items F such that $\sum_{i \in F} \vec{S}(i) \geq \vec{b}$. The value of $\vec{S}(i)$ is instantiated and fixed once we include item i in F. Once this decision is made, the item cannot be removed.*

We consider 4 classes of Stochastic Covering problems: (1) General Stochastic Covering where no restrictions are placed on item sizes or capacity; by scaling, we can assume that $\vec{b} = (1, 1, \ldots, 1)$. *(2) Stochastic Set Cover where the* $\vec{S}(i)$ *have values in* $\{0,1\}^d$ *and* $\vec{b} = (1, 1, \ldots, 1)$. *(3) Stochastic Covering with multiplicity where items can be selected repeatedly (i.e., each item is available with an unlimited number of copies). (4) Stochastic Set Cover with multiplicity where the* $\vec{S}(i)$ *have values in* $\{0,1\}^d$, $\vec{b} = (1, 1, \ldots, 1)$ *and in addition items can be selected repeatedly.*

We require a technical condition that the set of all items is feasible with probability 1. *For Stochastic Covering with multiplicity, it is sufficient to require that the set of all items is feasible with positive probability.*

When we refer to a stochastic optimization problem "with multiplicity", it means that each item on the input comes with an unlimited number of identical copies. This makes sense for deterministic PIP/CIP as well, where we could allow arbitrary integer vectors $\vec{x} \in \mathbb{Z}_+^n$. In the linear relaxation, we would then allow arbitrary non-negative vectors $\vec{x} \geq 0$. For example, the linear relaxation of a PIP with multiplicity would be

$$LP = \max \left\{ \vec{v} \cdot \vec{x} : \ A\vec{x} \leq \vec{b}, \ \vec{x} \in \mathbb{R}_+^n \right\}.$$

For all variants of stochastic optimization problems, we consider adaptive and non-adaptive policies.

Definition 10 (Adaptive and non-adaptive policies.)**.** *A non-adaptive policy is a fixed ordering of items to be inserted.*

An adaptive policy for a Stochastic Packing/Covering problem is a function $\mathcal{P} : 2^{[n]} \times \mathbb{R}_+^d \to [n]$. *The interpretation of* $\mathcal{P}$ *is that given a configuration* $(\mathcal{I}, \vec{b})$ *where* $\mathcal{I}$ *represents the items still available and* $\vec{b}$ *the remaining capacity,* $\mathcal{P}(\mathcal{I}, \vec{b})$ *determines which item should be chosen next among the items in* $\mathcal{I}$.

The value of the feasible solution found by a non-adaptive or adaptive policy is a random variable. Our objective function is the expectation of this variable, which we try to maximize in the case of packing problems, and minimize in the case of covering problems.

Regarding the information available a priori to a policy, whether adaptive or non-adaptive, we adopt the assumptions for Stochastic Knapsack stated in Section 5.1. That is, before inserting an item i, a policy has access only to the probability that the item alone fits, $\Pr[\vec{S}(i) \leq \vec{b}]$, and it has access to the mean truncated size of item i, defined below.

Definition 11. *For an instance with capacity* $\vec{b}$*, we define the mean truncated size of an item with random size* $\vec{S}$ *as a vector* $\vec{\mu}$ *such that*

$$\mu_j = \mathbf{E}[\min\{S_j, b_j\}].$$

5.3 Questions and gaps

Let's summarize the questions regarding stochastic optimization problems that we are interested in. In the following, "optimum" means "minimum" for covering problems and "maximum" for packing problems.

Adaptive policies represent strategies which could be implemented (computational efficiency aside) under the assumption that we can indeed detect the size of each item after its inclusion in the solution. Non-adaptive policies represent strategies which could be implemented even without this assumption. A question that we mentioned already is, what can be the benefit of being adaptive compared to being non-adaptive?

Another question that lends itself at this point is, how well could we perform if we knew the *precise sizes of all items* beforehand? This would be, in some sense, analogous to the analysis of on-line algorithms where we compare against an *omniscient* algorithm which has complete information about the input beforehand. In our setting, however, such an omniscient policy would be far too powerful.

Example. Consider an instance of Stochastic Knapsack: n items of unit value and size 2 $Be(1/2)$. I.e., any item overflows with probability 1/2 and adaptivity has no benefit. Any adaptive or non-adaptive policy inserts only 1 item on the average.

However, out of the total of n items, $n/2$ items on the average will attain size 0. An omniscient policy, knowing these items beforehand, can insert all of them, for an expected value of $n/2$.

This example demonstrates that even for Stochastic Knapsack, omniscient policies can be arbitrarily strong, compared to adaptive policies. In the following, we reject omniscient policies as unrealistically powerful and we do not consider them anymore. Our main focus is on the difference between adaptive and non-adaptive policies.

Definition 12. *For an instance of a stochastic packing/covering problem, define*

- $ADAPT$ = *optimum expected value achieved by an adaptive policy.*
- $NONADAPT$ = *optimum expected value achieved by a non-adaptive policy.*
- *The adaptivity gap is* $\alpha = ADAPT/NONADAPT$ *for packing problems, and* $\alpha = NONADAPT/ADAPT$ *for covering problems.*

For a class of stochastic packing/covering problems, we define α^* *as the supremum of possible adaptivity gaps.*

The adaptivity gap is our central concept. Note that it is defined in such a way that $\alpha^* \geq 1$ for both packing and covering problems. The greater the value of the adaptivity gap, the more advantage we can gain by making adaptive decisions. We investigate the question of how much benefit adaptivity can bring for each problem variant under consideration. As a motivating example, consider the Stochastic Knapsack problem.

Example.

- Item 1 has value $v_1 = 1$ and size $s_1 = Be(1/2)$.
- Item 2 has value $v_2 = 1$ and size $s_2 = 1$.
- Item 3 has value $v_3 = 1/p$ and size $s_3 = 2\ Be(p)$, for some small $p > 0$.

The optimal adaptive policy is: Insert item 1 first. If it has size 0, insert item 2, otherwise skip it. Finally insert item 3. The expected value achieved by this policy is $1 + 1/2 + 1 = 5/2$. On the other hand, the optimal fixed ordering of items can be seen to be $(1, 3, 2)$ which yields expected value $1 + p(1/p + 1/2) = 2 + p/2$. Thus the adaptivity gap can be arbitrarily close to $5/4$.

Surprisingly, this simple example is the best one we could find, and $5/4$ could possibly be the worst adaptivity gap for Stochastic Knapsack. However, proving upper bounds seems more challenging. An adaptive policy is basically a decision tree which can have potentially exponential size [14] so it is not very obvious how much power an adaptive policy can have. We defer this issue to Chapter 6.

Another notion that we introduce is the *randomness gap* which is motivated by the following:

Example. Consider two instances of Stochastic Knapsack. Let $p = 1/2 + \epsilon$ where $\epsilon > 0$. The first instance contains a large supply of items of random size $s_1 = Be(p)$. The second one has a large supply of items of deterministic size $s_2 = p$. All item values are 1. Note that the expected item sizes are p in both cases, however the expected values achieved by optimal adaptive policies are different. In the first instance, we can insert a sequence of items until two of them attain size 1; the expected number of trials until that happens is $2/p$, and $ADAPT_1 = 2/p - 1 = 3 - O(\epsilon)$. Whereas, only one item fits in the second instance: $ADAPT_2 = 1$. For small $\epsilon > 0$, the gap can be arbitrarily close to 3.

This example demonstrates that knowing only expected item sizes cannot determine the optimum to within a factor smaller than 3. Imagine that we prove an upper bound on $ADAPT$ in terms of expected item sizes. For our example, this upper bound would be at least 3; but it must be the same upper bound when applied to the second instance where the actual optimum is 1.

More generally, assume that two instances have the same mean sizes of items and in addition, truncated sizes are equal to non-truncated sizes; i.e. the truncated mean size $\vec{\mu}(i)$ is all the information the policy has about item i. Any bound on the performance of a policy that we prove in terms of mean item sizes must apply to both instances. Therefore the approximation factor we prove cannot be better than the ratio of actual optima for the two instances. The approximation factor we prove in such a way for a given class of problems cannot be better than the *randomness gap* for this class, defined below.

Definition 13. *For a pair of Stochastic Packing/Covering instances where every item fits within the capacity with probability* 1 *and the two instances become equal upon replacing item sizes* $\vec{S}(i)$ *by* $\vec{\mu}(i)$*, define*

- *Adaptive randomness gap:* $\rho_a = ADAPT_1/ADAPT_2$.
- *Non-adaptive randomness gap:* $\rho_n = NONADAPT_1/NONADAPT_2$.

For a class of stochastic optimization problems, we define ρ_a^* *and* ρ_n^* *as the suprema of* ρ_a *and* ρ_n *over all such pairs of instances.*

5.4 Literature review

5.4.1 Stochastic Optimization

Stochastic optimization problems have been investigated thoroughly in the operations research community. In particular, scheduling and packing problems have been treated in many different variants, under various constraints and also with random item sizes. This research dates back as far as 1966 [42]. In much of the previous literature, scheduling is studied with the objective to assign all given jobs to machines in order to minimize the completion time of the last job (or *makespan*). Sometimes, the weighted average of completion times is considered. Adaptivity is also a known concept in the stochastic programming literature. The distinction between adaptive and non-adaptive solutions is an important aspect of stochastic programming [5]. For a survey of stochastic scheduling problems, see [48].

Recently, stochastic optimization has come to the attention of the computer science community as well. An optimization model which has been mainly under scrutiny is the *two-stage stochastic optimization with recourse* [25, 21, 43]. In contrast to our model, this model involves only two stages of decision-making. In the first stage, only some information about the probability distribution of possible inputs is available. In the second stage, the precise input is known and the solution must be completed at any cost. The goal is to minimize the expected cost of the final solution. Another difference is that the randomness in this model is not in the properties of items forming a solution but rather in the demands to be satisfied by a solution. Let's illustrate this on the example of Set Cover: Shmoys and Swamy consider in [43] a Stochastic Set Cover problem where the sets to be chosen are deterministic and there is a random target set to be covered. In contrast, we consider a Stochastic Set Cover problem where the target set is fixed but the covering sets are random. This yields a setting of a very different flavor.

The knapsack problem has appeared in the literature in many forms and also with inherent randomness. Yet, perhaps surprisingly, the Stochastic Knapsack as we define it is a new problem and has not been studied in this form before.

5.4.2 Stochastic Knapsack

An aspect of Stochastic Knapsack which makes it different from traditional scheduling problems is the fixed deadline beyond which no profit is ever received. This issue has been addressed before: Derman, Lieberman and Ross [16] considered the adaptive stochastic knapsack problem where multiple copies of items are permitted, as well as the related *knapsack covering* problem in the same setting. An application of the knapsack covering problem is keeping a machine running for a certain duration, where the machine depends on a critical part (e.g. a light bulb) that periodically fails and must be replaced. The different items correspond to potential replacements, each having a deterministic cost and an uncertain lifetime. Derman et al. provide dynamic programming formulations for these problems, and also prove that if item sizes are exponentially distributed, both problems are solved by greedily scheduling jobs according to value or cost divided by expected size.

A different variant of Stochastic Knapsack, with deterministic sizes and random values has been studied by several authors [8, 23, 44, 47], all of whom consider the objective of computing a fixed set of items fitting in the knapsack that has maximum probability of achieving some target value (in this setting, maximizing expected value is equivalent to deterministic knapsack). Several heuristics have been proposed for this variant (e.g. branch-and-bound, preference-order dynamic programming), and results are given for specific probability distributions. Adaptivity is not considered by any of the authors. Another somewhat related variant, known as the stochastic and dynamic knapsack problem [30, 37], involves items that arrive on-line according to some stochastic process — we do not know the properties of an item until it arrives, at which point in time we must irrevocably decide to either accept the item and process it, or discard the item.

Two recent papers due to Kleinberg, Rabani and Tardos [29] and Goel and Indyk [20] consider yet another variant of the Stochastic Knapsack problem. Similarly to our model, they consider items with deterministic values and random sizes. However, their objective is quite different: given a specified overflow probability p, find a maximum-value set of items whose probability of overflowing the knapsack is at most p. Adaptivity is not considered. Kleinberg et al. consider only the case where item sizes have a Bernoulli-type distribution and for this case they provide a polynomial-time $O(\log 1/p)$-approximation algorithm as well as several pseudo-approximation results. For job sizes that have Poisson or exponential distributions, Goel and Indyk provide a PTAS, and for Bernoulli-distributed items they give an approximation scheme whose running time depends polynomially on n and $\log 1/p$. Kleinberg et al. show that the problem of computing the overflow probability of a set of items, even with Bernoulli distributions, is #P-hard. Consequently, it is also #P-hard to solve the problem variant mentioned above with deterministic sizes and random values, whose goal is maximizing the probability of achieving some specified target value.

In contrast to this work, our primary interest is the benefit of adaptivity and approximation of the adaptive optimum. Also, we do not assume any particular probability distributions. Our algorithms work for arbitrary random sizes.

5.4.3 Stochastic Packing

Stochastic Packing problems in this form have not been considered before. We build on the previous work on *Packing Integer Programs* (PIP) which was initiated by Raghavan and Thompson [40]. They proposed an LP approach combined with randomized rounding, which yields an $O(d)$-approximation for the general case [40]. For the case of set packing ($A \in \{0,1\}^{d\times n}, \vec{b} = (1,1,\ldots 1)$), their methods yield an $O(\sqrt{d})$-approximation. For general b parametrized by $B = \min b_i$, they get an $O(d^{1/B})$-approximation for $A \in [0,1]^{d\times n}$ and an $O(d^{1/(B+1)})$-approximation for $A \in \{0,1\}^{d\times n}$. One can also achieve a $\sqrt{d}$-approximation for the weighted set packing problem by a greedy algorithm [22].

This is complemented by the hardness results of Chekuri and Khanna [10], where it is shown that a $d^{1/(B+1)-\epsilon}$-approximation for the b-matching problem ($A \in \{0,1\}^{d\times n}, \vec{b} = (B,B,\ldots B)$) would imply $NP = ZPP$ (using Håstad's result on the inapproximability of Max Clique [46]). The analysis of the randomized rounding technique has been refined by Srinivasan [45] who presents stronger bounds; however, the approximation factors are not improved in general (and in the case of $A \in \{0,1\}^{d\times n}$, this is essentially impossible due to [10]). A remaining question was the gap between $d^{1/2-\epsilon}$ and $O(d)$ in the general case, as well as the gap between $d^{1/(B+1)-\epsilon}$ and $O(d^{1/B})$ in case $A \in [0,1]^{d\times n}, \vec{b} = (B,B,\ldots,B)$.

5.4.4 Stochastic Covering

Stochastic Covering problems can be seen as generalizations of *Covering Integer Programs* (CIP, see [45]). The forefather of Covering Integer Programs is the well-known Set Cover problem. For Set Cover, it was proved by Johnson [27] that the greedy algorithm gives an approximation factor of $\ln d$. This result was extended by Chvátal to the weighted case [11]. The same approximation guarantee can be obtained by a linear programming approach, as shown by Lovász [32]. Finally, it was proved by Uriel Feige [17] that these results are optimal, in the sense that a polynomial-time $(1-\epsilon)\ln d$ approximation algorithm for Set Cover would imply $NP \subset TIME(n^{O(\log\log n)})$.

Note. Usually the cardinality of the ground set is denoted by n, but to be consistent with Stochastic Packing problems, we view this parameter as "dimension" and denote it by d.

For general Covering Integer Problems, the optimal approximation has been found only recently - see [7], [31]. The approximation factor turns out to be again $O(\log d)$ but the approximation algorithm is more sophisticated since the natural LP can have an arbitrarily large integrality gap (see also Section 8.2).

Chapter 6

Stochastic Knapsack

In this chapter, we address the Stochastic Knapsack problem. Recall that there is an example for Stochastic Knapsack where the adaptivity gap is arbitrarily close to 5/4 (Section 5.3). Surprisingly, this simple example is the best one we could find, and 5/4 could possibly be the worst adaptivity gap for Stochastic Knapsack. However, proving upper bounds is much more challenging. Our efforts have led to a sequence of improvements on the adaptivity gap factor, starting from 50 and culminating (so far) in the bound of 4 presented in this thesis. We show two approaches in this thesis: the first one leading to a bound of 32/7 and the second one proving the bound of 4.

6.1 Bounding adaptive policies

A fundamental problem is, how much value can an adaptive policy achieve? In this section, we provide a non-trivial upper bound on the performance of any adaptive policy for Stochastic Knapsack. In [15], this bound was proved using a martingale argument. Here, we give an alternative elementary proof.

Let's recall that the random sizes of items are denoted by $s_1, s_2, \ldots, s_n$ and their mean truncated sizes by $\mu_i = \mathbf{E}[\min\{s_i, 1\}]$. For now, we ignore item values and analyze the *mean truncated size* $\mu(\mathcal{A}) = \sum_{i \in \mathcal{A}} \mu_i$ of all the items that an adaptive policy attempts to insert. (For a set of items $\mathcal{A}$, we refer to the quantity $\mu(\mathcal{A})$ as "the mass of A".) I.e., we count the mass of all items inserted including the first overflowing item.

Lemma 15. *For a Stochastic Knapsack instance with capacity* 1 *and any adaptive policy, let* $\mathcal{A}$ *denote the (random) set of items which the policy attempts to insert. Then*

$$\mathbf{E}[\mu(\mathcal{A})] \leq 2.$$

Proof. Denote by $M(c)$ the maximum possible $\mathbf{E}[\mu(\mathcal{A})]$ for a random set $\mathcal{A}$ that an adaptive policy can attempt to insert within remaining space $c \leq 1$. We prove, by induction on the number of available items, that $M(c) \leq 1 + c$.

Suppose that an optimal adaptive policy, given remaining space c, inserts item i. Denote by $fit(i, c)$ the characteristic function of the event that item i fits ($s_i \leq c$) and

by $\tilde{s}_i$ its truncated size $\tilde{s}_i = \min\{s_i, 1\}$. Having inserted item i, the policy continues only if item i fits and then the remaining space is $c - \tilde{s}_i \geq 0$:

$$M(c) = \mu_i + \mathbf{E}[fit(i,c)M(c - \tilde{s}_i)] = \mathbf{E}[\tilde{s}_i + fit(i,c)M(c - \tilde{s}_i)].$$

Applying the induction hypothesis to $M(c - \tilde{s}_i)$,

$$\begin{aligned} M(c) &\leq \mathbf{E}[\tilde{s}_i + fit(i,c)(1 + c - \tilde{s}_i)] \\ &= \mathbf{E}[fit(i,c)(1+c) + (1 - fit(i,c))\ \tilde{s}_i] \\ &\leq \mathbf{E}[fit(i,c)(1+c) + (1 - fit(i,c))] \\ &\leq 1 + c. \end{aligned}$$

□

An adaptive policy can take decisions based on the perceived sizes of items; nonetheless, the total probability that an item i is inserted by the policy is determined beforehand - as an average over all the branches of the decision tree where item i is inserted, weighted by the probabilities of executing those branches (which are determined by the policy and distributions of item sizes). We do not actually need to write out this probability explicitly in terms of the policy. Just denote by x_i the total probability that the policy attempts to insert item i. Also, define by $w_i = v_i\ \Pr[s_i \leq 1]$ the *effective item value* of item i which is an upper bound on the expected profit a policy can gain if it attempts to insert item i. Then we get the following bound.

Theorem 15. *For Stochastic Knapsack,*

$$ADAPT \leq \Phi(2)$$

where

$$\Phi(t) = \max\left\{\sum_i w_i x_i : \sum_i x_i \mu_i \leq t,\ x_i \in [0,1]\right\}$$

and $w_i = v_i\ \Pr[s_i \leq 1]$, $\mu_i = \mathbf{E}[\min\{s_i, 1\}]$.

Proof. Consider an adaptive policy which attempts to insert item i with total probability x_i. Conditioned on inserting item i, the expected profit received for it can be at most w_i. So the expected value achieved by the policy is bounded by $\sum_i w_i x_i$. The expected mass that the policy attempts to insert is $\mathbf{E}[\mu(\mathcal{A})] = \sum_i x_i \mu_i$. We know that this is bounded by $\mathbf{E}[\mu(\mathcal{A})] \leq 2$, therefore $\vec{x}$ is a feasible vector for $\Phi(2)$ and the expected value obtained is at most $\sum_i w_i x_i \leq \Phi(2)$. □

Note that $\Phi(t)$ is a fractional solution of the knapsack problem with capacity t and mean truncated item sizes. This fact will be useful when we design a non-adaptive policy to approximate the adaptive optimum.

6.2 A 32/7-approximation for Stochastic Knapsack

Consider the function $\Phi(t)$ which can be seen as the fractional solution of a knapsack problem with capacity t. This function is easy to describe. Its value is achieved by packing items of maximum possible "value density" and taking a suitable fraction of the overflowing item. Assume that the items are already indexed by decreasing value density:

$$\frac{w_1}{\mu_1} \geq \frac{w_2}{\mu_2} \geq \frac{w_3}{\mu_3} \geq \dots.$$

We call this the *greedy ordering*. Also, let $M_k = \sum_{i=1}^{k} \mu_i$. Then for $t = M_{k-1} + \xi \in [M_{k-1}, M_k]$, we have

$$\Phi(t) = \sum_{i=1}^{k-1} w_i + \frac{\xi}{\mu_k} w_k.$$

Assume WLOG that $\Phi(1) = 1$. This can be arranged by scaling the values by a constant factor which doesn't change the adaptivity gap. We also assume that there are sufficiently many items so that $\sum_{i=1}^{n} \mu_i \geq 1$ which can be arranged by adding dummy items of value 0. Now we are ready to describe our algorithm.

The randomized greedy algorithm.

Let r be the minimum index such that $\sum_{i=1}^{r} \mu_i \geq 1$. Denote $\mu'_r = 1 - \sum_{i=1}^{r-1} \mu_i$, i.e. the part of μ_r that fits within capacity 1. Set $p' = \mu'_r/\mu_r$ and $w'_r = p'w_r$. For $j = 1, 2, \dots, r-1$, set $w'_j = w_j$ and $\mu'_r = \mu_r$. We assume $\Phi(1) = \sum_{i=1}^{r} w'_i = 1$.

- Choose index k with probability w'_k.
- If $k < r$, insert item k. If $k = r$, flip another independent coin $Be(p')$ and insert item r only in case of success (otherwise discard it).
- Then insert $1, 2, \dots, k-1, k+1, \dots, r$ in the greedy order.

Theorem 16. *The randomized greedy algorithm achieves expected value*

$$GREEDY \geq \frac{7}{32}\ ADAPT.$$

Proof. First, assume for simplicity that $\sum_{i=1}^{r} \mu_i = 1$. Also, $\Phi(1) = \sum_{i=1}^{r} w_i = 1$. Then $ADAPT \leq \Phi(2) \leq 2$ but also, more strongly:

$$ADAPT \leq \Phi(2) \leq 1 + \omega$$

where $\omega = w_r/\mu_r$. This follows from the concavity of $\Phi(x)$. Note that

$$\omega = \frac{w_r}{\mu_r} \leq \frac{\sum_{i=1}^{r} w_i}{\sum_{i=1}^{r} \mu_i} = 1.$$

With $\sum_{i=1}^{r} \mu_i = 1$, the algorithm has a simpler form:

- Choose $k \in \{1, 2, \ldots, r\}$ with probability w_k and insert item k first.
- Then, insert $1, 2, \ldots, k-1, k+1, \ldots, r$ in the greedy order.

We estimate the expected value achieved by this algorithm. Note that we analyze the expectation with respect to the random sizes of items and also our own randomization. Item k is inserted with probability w_k first, with probability $\sum_{i=1}^{k-1} w_i$ after $\{1, 2, \ldots, k-1\}$ and with probability w_j after $\{1, 2, \ldots, k-1, j\}$ (for $k < j \leq r$). If it's the first item, the expected profit for it is simply $w_k = v_k \cdot \Pr[s_k \leq 1]$. Otherwise we use Markov's inequality:

$$\Pr\left[\sum_{i=1}^{k} s_i \leq 1\right] \geq 1 - \sum_{i=1}^{k} \mu_i.$$

The case when item k is preceded by $\{1, 2, \ldots, k-1, j\}$ is similar. Let V_k denote our lower bound on the expected profit obtained for item k:

$$\begin{aligned} V_k &= w_k \left(w_k + \sum_{j=1}^{k-1} w_j \left(1 - \sum_{i=1}^{k} \mu_i\right) + \sum_{j=k+1}^{r} w_j \left(1 - \sum_{i=1}^{k} \mu_i - \mu_j\right)\right) \\ &= w_k \left(\sum_{j=1}^{r} w_j \left(1 - \sum_{i=1}^{k} \mu_i\right) + w_k \sum_{i=1}^{k} \mu_i - \sum_{j=k+1}^{r} w_j \mu_j \right). \end{aligned}$$

Let $V = \sum_{k=1}^{r} V_k$ and simplify V using $\sum_{j=1}^{r} w_j = 1$ and $\sum_{j=1}^{r} \mu_j = 1$:

$$\begin{aligned} V &= \sum_{k=1}^{r} w_k \left(1 - \sum_{i=1}^{k} \mu_i + w_k \sum_{i=1}^{k} \mu_i - \sum_{i=k+1}^{r} w_i \mu_i\right) \\ &= 1 + \sum_{i \leq k \leq r} (-w_k \mu_i + w_k^2 \mu_i) - \sum_{k < i \leq r} w_k w_i \mu_i \\ &= 1 + \sum_{i \leq k \leq r} (-w_k \mu_i + w_k^2 \mu_i + w_k w_i \mu_i) - \sum_{i,k=1}^{r} w_k w_i \mu_i \\ &= 1 + \sum_{i \leq k \leq r} w_k \mu_i (w_i + w_k - 1) - \sum_{i=1}^{r} w_i \mu_i. \end{aligned}$$

To symmetrize this polynomial, we apply the condition of greedy ordering. For any $i < k$, we have $w_i + w_k - 1 \leq 0$, and the ordering implies $w_k \mu_i \leq w_i \mu_k$ which allows us to replace $w_k \mu_i$ by $\frac{1}{2}(w_k \mu_i + w_i \mu_k)$ for all pairs $i < k$:

$$\begin{aligned} V &\geq 1 + \frac{1}{2} \sum_{i < k \leq r} (w_k \mu_i + w_i \mu_k)(w_i + w_k - 1) + \sum_{i=1}^{r} w_i \mu_i (2w_i - 1) - \sum_{i=1}^{r} w_i \mu_i \\ &= 1 + \frac{1}{2} \sum_{i,k=1}^{r} w_k \mu_i (w_i + w_k - 1) + \frac{1}{2} \sum_{i=1}^{r} w_i \mu_i (2w_i - 1) - \sum_{i=1}^{r} w_i \mu_i \end{aligned}$$

and using again $\sum_{j=1}^{r} w_j = \sum_{j=1}^{r} \mu_j = 1$,

$$\begin{aligned} V &\geq 1 + \frac{1}{2}\sum_{i=1}^{r} \mu_i(w_i - 1) + \sum_{k=1}^{r} w_k^2 + \sum_{i=1}^{r} w_i^2\mu_i - \frac{3}{2}\sum_{i=1}^{r} w_i\mu_i \\ &= \frac{1}{2} + \frac{1}{2}\sum_{k=1}^{r} w_k^2 + \sum_{i=1}^{r} w_i^2\mu_i - \sum_{i=1}^{r} w_i\mu_i. \end{aligned}$$

We want to compare this expression to $1 + \omega$ where $\omega = \min_{i \leq r} w_i/\mu_i$. We use the value of ω to estimate $\sum_{k=1}^{r} w_k^2 \geq \omega \sum_{k=1}^{r} w_k\mu_k$ and we get

$$\begin{aligned} V &\geq \frac{1}{2} + \frac{\omega}{2}\sum_{k=1}^{r} w_k\mu_k + \sum_{i=1}^{r} w_i^2\mu_i - \sum_{i=1}^{r} w_i\mu_i \\ &= \frac{1}{2} + \sum_{i=1}^{r} w_i\mu_i\left(\frac{\omega}{2} + w_i - 1\right) \geq \frac{1}{2} - \sum_{i=1}^{r}\left(\frac{1}{2} - \frac{\omega}{4}\right)^2 \mu_i \end{aligned}$$

by minimizing a quadratic function in w_i for each i. Finally, $\sum_i \mu_i = 1$ and

$$V \geq \frac{1}{4} + \frac{\omega}{4} - \frac{\omega^2}{16}.$$

We compare this to the adaptive optimum which is bounded by $1 + \omega$:

$$\frac{V}{1+\omega} \geq \frac{1}{4} - \frac{\omega^2}{16(1+\omega)}$$

which is at least $7/32$ for any $\omega \in [0, 1]$.

It remains to remove the assumption that $\sum_{i=1}^{r} \mu_i = 1$. We claim that if $\sum_{i=1}^{r} \mu_i > 1$, the randomized greedy algorithm performs just like the same algorithm on a modified instance with values w'_j and mean sizes μ'_j (see the description of the algorithm). Indeed, $\Phi(1) = 1$ and $\omega = w_r/\mu_r = w'_r/\mu'_r$ in both cases, so the bound on $ADAPT$ is the same. For an item $k < r$, our estimate of the expected profit in both instances is

$$V_k = w'_k\left(w'_k + \sum_{j=1}^{k-1} w'_j\left(1 - \sum_{i=1}^{k}\mu'_i\right) + \sum_{j=k+1}^{r} w'_j\left(1 - \sum_{i=1}^{k}\mu'_i - \mu'_j\right)\right).$$

For the original instance, this is because the expected contribution of item r to the total size, conditioned on being selected first, is $p'\mu_r = \mu'_r$; if not selected first, its contribution is not counted at all. The expected profit for item r is $V_r = w'_r p' w_r = (w'_r)^2$ in both instances. This reduces the analysis to the case we dealt with already.

Finally, the expected value obtained by our randomized policy is a convex combination of expected values obtained by *deterministic* non-adaptive policies: Insert a selected item first, and then follow the greedy ordering. We can estimate the expected value for each such ordering in polynomial time and then choose the best one, which must achieve at least $7/32$ $ADAPT$. □

Example 1. The analysis is tight in the following sense: Consider an instance of 8 equal items with $\mu_i = 1/4$ and $w_i = v_i = 1$. Our bound on the adaptive optimum would be $\Phi(2) = 8$, while our analysis of any non-adaptive algorithm would imply the following. We get the first item always (because $w_1 = v_1 = 1$), the second one with probability at least $1 - 2/4 = 1/2$ and the third one with probability at least $1 - 3/4 = 1/4$. Thus our estimate of the expected value obtained is $7/4$. We cannot prove a better bound than $32/7$ with the tools we are using: the LP from Theorem 19, and Markov bounds based on mean item sizes. Of course, the actual adaptivity gap for this instance is 1, and our algorithm performs optimally.

Example 2. Our randomized greedy algorithm can be as bad as $ADAPT/4$. Consider for example items of size $s_1 = (1+\epsilon)/2$ and value $v_1 = 1/2 + \epsilon$, while the other type of item is $s_2 = Be(p), v_2 = p$. Our algorithm will choose a sequence of items of the first type, of which only one can fit. The optimum is a sequence of items of the second type which yields expected value $2 - p$. For small $p, \epsilon > 0$, the gap can be arbitrarily close to 4. We have no example where the greedy algorithm performs worse than this. The approximation factor we can prove is $32/7 \doteq 4.57$ but it seems that the gap between 4 and 4.57 is only due to the weakness of Markov bounds.

Example 3. Even with a different non-adaptive algorithm, we cannot prove a bound better than 4 using this LP. The limitation is that the actual gap between $\Phi(2)$ and $ADAPT$ can be arbitrarily close to 4: Consider items of deterministic size $(1+\epsilon)/2$ for a small $\epsilon > 0$. Fractionally, we can pack almost 4 items within capacity 2, so that $\Phi(2) = 4/(1+\epsilon)$, while only 1 item can actually fit. This means that no approximation bound based on the LP from Theorem 19, adaptive or non-adaptive, can break the barrier of 4.

6.3 A stronger bound on adaptive policies

We aim to improve the upper bound on the adaptivity gap to 4. The randomized greedy algorithm from Section 6.2 might actually achieve this approximation ratio, but we cannot prove it comparing to the bound of $\Phi(2)$ developed in Section 6.1. Here we strengthen this upper bound on adaptive policies. Then even a simpler greedy algorithm will suffice to prove the approximation factor of 4.

Denote by $\mathcal{A}$ the set of items that an adaptive policy attempts to insert. In general, we know that $\mathbf{E}[\mu(\mathcal{A})] \leq 2$. Here, we examine more closely how this mass can be distributed among items. Assume that $\mathcal{J}$ is a subset of items and $\mathcal{B} \subseteq \mathcal{J}$ is a (random) subset of these items that a policy attempts to insert. How much mass can $\mathcal{B}$ possibly have? The bound on $\mathbf{E}[\mu(\mathcal{B})]$ approaches 2 asymptotically as $\mu(\mathcal{J}) \to \infty$ but we get a stronger bound for small $\mu(\mathcal{J})$.

Lemma 16. *Let $\mathcal{B}$ be the subset of $\mathcal{J}$ that an adaptive policy tries to insert. Then*

$$\mathbf{E}[\mu(\mathcal{B})] \leq 2\left(1 - \prod_{j \in \mathcal{J}}(1-\mu_j)\right).$$

Proof. Denote by $M(\mathcal{J}, c)$ the maximum possible expected mass $\mathbf{E}[\mu(\mathcal{B})]$ that an adaptive policy can attempt to insert within capacity c, using only items $\mathcal{B} \subseteq \mathcal{J}$. We prove by induction on $|\mathcal{J}|$ that

$$M(\mathcal{J}, c) \leq (1 + c) \left(1 - \prod_{j \in \mathcal{J}} (1 - \mu_j) \right).$$

Consider that in a given configuration $(\mathcal{J}, c)$, a policy optimal with respect to our objective inserts item i. The policy will collect mass μ_i and then continue provided that $s_i \leq c$. We denote the characteristic function of this event as "$fit(i, c)$", and the set of remaining items as $\mathcal{J}' = \mathcal{J} \setminus \{i\}$.

$$M(\mathcal{J}, c) = \mu_i + \mathbf{E}[fit(i, c) M(\mathcal{J}', c - s_i)].$$

We apply the induction hypothesis to $M(\mathcal{J}', c - s_i)$:

$$M(\mathcal{J}, c) \leq \mu_i + \mathbf{E} \left[fit(i, c)(1 + c - s_i) \left(1 - \prod_{j \in \mathcal{J}'} (1 - \mu_j) \right) \right].$$

We denote the truncated size of item i by $\tilde{s}_i = \min\{s_i, 1\}$. Conditioned on $fit(i, c)$, we can replace s_i by $\tilde{s}_i$:

$$M(\mathcal{J}, c) \leq \mu_i + \mathbf{E} \left[fit(i, c)(1 + c - \tilde{s}_i) \left(1 - \prod_{j \in \mathcal{J}'} (1 - \mu_j) \right) \right].$$

and then we note that $1 + c - \tilde{s}_i \geq 0$ holds always, not only when item i fits. So we can replace $fit(i, c)$ by 1 and evaluate the expectation:

$$\begin{aligned} M(\mathcal{J}, c) &\leq \mu_i + \mathbf{E} \left[(1 + c - \tilde{s}_i) \left(1 - \prod_{j \in \mathcal{J}'} (1 - \mu_j) \right) \right] \\ &= (1 + c) - (1 + c - \mu_i) \prod_{j \in \mathcal{J}'} (1 - \mu_j). \end{aligned}$$

Finally, using $(1 + c - \mu_i) \geq (1 + c)(1 - \mu_i)$, we conclude:

$$M(\mathcal{J}, c) \leq (1 + c) - (1 + c)(1 - \mu_i) \prod_{j \in \mathcal{J}'} (1 - \mu_j) = (1 + c) \left(1 - \prod_{j \in \mathcal{J}} (1 - \mu_j) \right).$$

□

How does this restrict the expected value achieved by the optimal adaptive policy? Denoting by x_i the total probability that an adaptive policy attempts to insert item i, we can write an LP similar to Theorem 15.

$$\Psi(t) = \max\left\{ \sum_i w_i x_i : \begin{array}{l} \forall \mathcal{J} \subseteq \mathcal{I};\ \sum_{i\in\mathcal{J}} \mu_i x_i \le t(1 - \prod_{j\in\mathcal{J}}(1-\mu_j)) \\ \forall i \in \mathcal{I};\ x_i \in [0,1] \end{array} \right\}.$$

Then $ADAPT \le \Psi(2)$ because the probabilities x_i corresponding to any adaptive policy form a feasible solution for $\Psi(2)$. This is a strengthening of Theorem 15 in the sense that any solution feasible for $\Psi(2)$ is feasible for $\Phi(2)$. Therefore $\Psi(2) \le \Phi(2)$.

However, for now we disregard the condition that $x_i \le 1$. This yields a simplified upper bound which is easier to handle and sufficient to prove the bound of 4. Denote by $w_i = v_i \ \Pr[s_i \le 1]$ the effective item values and order items as before:

$$\frac{w_1}{\mu_1} \ge \frac{w_2}{\mu_2} \ge \frac{w_3}{\mu_3} \ge \ldots \ge \frac{w_n}{\mu_n}.$$

Theorem 17.

$$ADAPT \le 2\sum_{k=1}^{n} w_k \prod_{i=1}^{k-1}(1-\mu_i).$$

Proof. Consider any optimal policy. Let x_i denote the probability that it attempts to insert item i. Then $ADAPT \le \sum_{i=1}^{n} w_i x_i$. We set formally $w_{n+1}/\mu_{n+1} = 0$ and rewrite this sum as

$$ADAPT \le \sum_{i=1}^{n} \frac{w_i}{\mu_i}\mu_i x_i = \sum_{k=1}^{n}\left(\frac{w_k}{\mu_k} - \frac{w_{k+1}}{\mu_{k+1}}\right)\sum_{i=1}^{k}\mu_i x_i$$

where all the summands are non-negative. Denote by $\mathcal{B}_k$ the subset of the first k items that the policy attempts to insert before overflowing. By Lemma 16, we know that

$$\sum_{i=1}^{k}\mu_i x_i = \mathbf{E}[\mu(\mathcal{B}_k)] \le 2\left(1 - \prod_{i=1}^{k}(1-\mu_i)\right)$$

and so

$$\begin{aligned} ADAPT &\le 2\sum_{k=1}^{n}\left(\frac{w_k}{\mu_k} - \frac{w_{k+1}}{\mu_{k+1}}\right)\left(1 - \prod_{i=1}^{k}(1-\mu_i)\right) \\ &= 2\sum_{k=1}^{n}\frac{w_k}{\mu_k}\left(\prod_{i=1}^{k-1}(1-\mu_i) - \prod_{i=1}^{k}(1-\mu_i)\right) \\ &= 2\sum_{k=1}^{n} w_k \prod_{i=1}^{k-1}(1-\mu_i). \end{aligned}$$

□

6.4 A 4-approximation for Stochastic Knapsack

We assume that items are ordered so that

$$\frac{w_1}{\mu_1} \geq \frac{w_2}{\mu_2} \geq \frac{w_3}{\mu_3} \geq \ldots \geq \frac{w_n}{\mu_n}$$

Based on this ordering, we define a simple greedy algorithm.

The vanilla greedy algorithm.

- Let $V = \sum_{k=1}^{n} w_k \prod_{i=1}^{k-1}(1-\mu_i)$.
- If there is an item such that $w_i \geq V/2$, insert only this item.
- Otherwise, insert all items in the greedy order.

We claim that this algorithm achieves expected value at least $V/2$. First, we prove a general lemma on sums of random variables. The lemma estimates the expected mass that our algorithm *attempts* to insert.

Lemma 17. *Let $X_1, X_2, \ldots, X_k$ be independent, nonnegative random variables and $\mu_i = \mathbf{E}[\min\{X_i, 1\}]$. Let $S_0 = 0$ and $S_{i+1} = S_i + X_{i+1}$. Let A_i be the event that $S_i \leq 1$ and $p_i = \Pr[A_i]$. Then*

$$\sum_{j=1}^{k} p_{j-1}\mu_j \geq 1 - \prod_{j=1}^{k}(1-\mu_j).$$

Note. We need not assume anything about the total expectation. This works even for $\sum_{i=1}^{k} \mu_i > 1$.

For the special case of k random variables of equal expectation $\mu_j = 1/k$, we get

$$\sum_{j=1}^{k} \Pr[S_{j-1} \leq 1] \geq k\left(1 - \left(1 - \frac{1}{k}\right)^k\right) \geq k\left(1 - \frac{1}{e}\right). \tag{6.1}$$

This seems related to a question raised by Uriel Feige [18]: What is the probability that $S_{k-1} \leq 1$ for a sum of independent random variables $S_{k-1} = X_1 + X_2 + \ldots + X_{k-1}$ with expectations $\mu_j = 1/k$? Feige proves that the probability is at least $1/13$ but conjectures that it is in fact at least $1/e$. A more general conjecture would be that for any $j < k$,

$$p_j = \Pr[S_j \leq 1] \geq \left(1 - \frac{1}{k}\right)^j \tag{6.2}$$

Note that Markov's inequality would give only $p_j \geq 1 - j/k$. Summing up (6.2) from $j = 0$ up to $k-1$, we would get (6.1). However, (6.2) remains a conjecture and we can only prove (6.1) as a special case of Lemma 17.

Proof. Define $\sigma_i = \mathbf{E}[S_i|A_i]$ where A_i is the event that $S_i \leq 1$. First, assume $\sigma_i + \mu_{i+1} < 1$ for all i, $0 \leq i < k$. By conditional expectations,

$$\begin{aligned}
\sigma_i + \mu_{i+1} &= \mathbf{E}[S_i + \min\{X_{i+1}, 1\}|A_i] = \\
&= \mathbf{E}[S_{i+1}|A_{i+1}] \Pr[A_{i+1}|A_i] + \mathbf{E}[S_i + \min\{X_{i+1}, 1\}|\bar{A}_{i+1}] \Pr[\bar{A}_{i+1}|A_i] \\
&\geq \sigma_{i+1} \frac{\Pr[A_{i+1}]}{\Pr[A_i]} + 1 \cdot \left(1 - \frac{\Pr[A_{i+1}]}{\Pr[A_i]}\right) \\
&= \sigma_{i+1} \frac{p_{i+1}}{p_i} + \left(1 - \frac{p_{i+1}}{p_i}\right) \\
&= 1 - (1 - \sigma_{i+1}) \frac{p_{i+1}}{p_i}.
\end{aligned}$$

This implies that

$$\frac{p_{i+1}}{p_i} \geq \frac{1 - \sigma_i - \mu_{i+1}}{1 - \sigma_{i+1}}. \tag{6.3}$$

For $i = 0$, we get $p_1 \geq (1 - \mu_1)/(1 - \sigma_1)$, since $p_0 = 1$ and $\sigma_0 = 0$. Multiplying (6.3) from $i = 0$ up to $j - 1$, we get

$$\begin{aligned}
p_j &\geq \frac{1 - \mu_1}{1 - \sigma_1} \cdot \frac{1 - \sigma_1 - \mu_2}{1 - \sigma_2} \cdots \frac{1 - \sigma_{j-1} - \mu_j}{1 - \sigma_j} \\
&= (1 - \mu_1) \left(1 - \frac{\mu_2}{1 - \sigma_1}\right) \cdots \left(1 - \frac{\mu_j}{1 - \sigma_{j-1}}\right) \frac{1}{1 - \sigma_j}.
\end{aligned}$$

Let's define

$$\nu_i = \frac{\mu_i}{1 - \sigma_{i-1}}.$$

Due to our assumptions, $\mu_i \leq \nu_i \leq 1$; as a special case, $\nu_1 = \mu_1$. I.e.,

$$p_j \geq \frac{1}{1 - \sigma_j} \prod_{i=1}^{j} (1 - \nu_i), \tag{6.4}$$

and

$$\sum_{j=1}^{k} p_{j-1}\mu_j \geq \sum_{j=1}^{k} \nu_j \prod_{i=1}^{j-1} (1 - \nu_i) = 1 - \prod_{i=1}^{k} (1 - \nu_i). \tag{6.5}$$

Since $\nu_i \geq \mu_i$, we get

$$\sum_{j=1}^{k} p_{j-1}\mu_j \geq 1 - \prod_{i=1}^{k} (1 - \mu_i). \tag{6.6}$$

We have proved the statement of the lemma, provided that $\sigma_i + \mu_{i+1} < 1$ for all $i < k$. Finally, we handle the case when for some $j < k$, $\sigma_j + \mu_{j+1} \geq 1$; consider the first such j. Then $\sigma_i + \mu_{i+1} < 1$ for all $i < j$ and we can apply the previous arguments to

variables $X_1, \ldots, X_j$. From (6.5),

$$\sum_{i=1}^{j} p_{i-1}\mu_i \geq 1 - \prod_{i=1}^{j}(1-\nu_i). \tag{6.7}$$

In addition, we estimate the contribution of the $(j+1)$-th item, which has mass $\mu_{j+1} \geq 1 - \sigma_j$, and from (6.4) we get

$$p_j\mu_{j+1} \geq p_j(1-\sigma_j) \geq \prod_{i=1}^{j}(1-\nu_i). \tag{6.8}$$

Therefore in this case we get from (6.7) + (6.8):

$$\sum_{i=1}^{k} p_{i-1}\mu_i \geq \sum_{i=1}^{j} p_{i-1}\mu_i + p_j\mu_{j+1} \geq 1.$$

□

Theorem 18. *The expected value obtained by the vanilla greedy algorithm is*

$$GREEDY^* \geq \frac{V}{2} \geq \frac{ADAPT}{4}.$$

Proof. Let $X_i = s_i$ be the random size of item i. Lemma 17 says that the expected mass that our greedy algorithm *attempts to insert*, restricted to the first k items, is at least $1 - \prod_{i=1}^{k}(1-\mu_i)$. As in Lemma 17, we denote by p_k the probability that the first k items fit. Now we estimate the "virtual value" obtained, which is the following: for each item i that the algorithm attempts to insert, we count a contribution of w_i. Since item i is inserted with probability p_{i-1}, this would be

$$\sum_{i=1}^{n} p_{i-1}w_i = \sum_{i=1}^{n} \frac{w_i}{\mu_i} p_{i-1}\mu_i = \sum_{k=1}^{n} \left(\frac{w_k}{\mu_k} - \frac{w_{k+1}}{\mu_{k+1}}\right) \sum_{i=1}^{k} p_{i-1}\mu_i.$$

Using Lemma 17,

$$\begin{aligned}
\sum_{i=1}^{n} p_{i-1}w_i &\geq \sum_{k=1}^{n} \left(\frac{w_k}{\mu_k} - \frac{w_{k+1}}{\mu_{k+1}}\right) \left(1 - \prod_{i=1}^{k}(1-\mu_i)\right) \\
&= \sum_{k=1}^{n} \frac{w_k}{\mu_k} \left(\prod_{i=1}^{k-1}(1-\mu_i) - \prod_{i=1}^{k}(1-\mu_i)\right) \\
&= \sum_{k=1}^{n} \frac{w_k}{\mu_k} \left(\prod_{i=1}^{k-1}(1-\mu_i)\right)(1-(1-\mu_k)) \\
&= \sum_{k=1}^{n} w_k \prod_{i=1}^{k-1}(1-\mu_i) = V.
\end{aligned}$$

Note that this is exactly 1/2 of the upper bound on the "virtual value" obtained by the optimal adaptive policy (Theorem 17). If there is an item such that $w_i \geq V/2$ then inserting this item alone yields expected value at least $ADAPT/4$. So we can assume that for every i, $w_i < V/2$. Then the difference between the virtual and actual value obtained by the greedy algorithm is

$$\sum_{i=1}^{n} (p_{i-1} - p_i) w_i \leq \frac{V}{2} \sum_{i=1}^{n} (p_{i-1} - p_i) \leq \frac{V}{2}.$$

We conclude that

$$GREEDY^* \geq \sum_{i=1}^{n} p_i w_i \geq \sum_{i=1}^{n} p_{i-1} w_i - \frac{V}{2} \geq \frac{V}{2} \geq \frac{ADAPT}{4}.$$

□

Example 1. The analysis of the vanilla greedy algorithm is tight. Recall Example 2 from Section 6.2. The vanilla greedy algorithm will perform just like the randomized greedy algorithm, which obtains close to 1/4 of the adaptive optimum.

Example 2. Using Theorem 17 to bound the adaptive optimum, we cannot prove a better approximation factor than 4. For an instance with an unlimited supply of items of value $v_i = 1$ and deterministic size $s_i = (1+\epsilon)/2$, we get $V = 2/(1+\epsilon)$ and therefore an upper bound of $4/(1+\epsilon)$ on $ADAPT$, while only 1 item can fit.

The same holds even for the stronger bound of $\Psi(2)$ (Section 6.3). Since $x_i = \min\{1, 2^{3-i}\}/(1+\epsilon)$ is a feasible solution whose value converges to $\sum_{i=1}^{\infty} x_i = 4/(1+\epsilon)$, we get $\Psi(2) \geq 4/(1+\epsilon)$.

Example 3. As we mentioned in Section 5.3 already, there are two instances with items of the same mean size (either $s_i = Be(1/2+\epsilon)$ or $s'_i = 1/2+\epsilon$) such that the ratio of the respective adaptive optima is close to 3. Another way to look at this example is that in an instance where both item types are available, the optimum is close to 3 (inserting a sequence of $Be(1/2+\epsilon)$ items), but an algorithm which has only access to expected item sizes has no way to distinguish the two types of items. It may as well choose a sequence of items of size $1/2+\epsilon$ and then only 1 item fits. This means that with the knowledge of only mean truncated item sizes, no approximation algorithm, adaptive or non-adaptive, can surpass the barrier of 3.

Chapter 7

Stochastic Packing

In this chapter, we focus on Stochastic Packing problems. We shall consider this class of problems in its full generality, as well as some restricted classes of problems. For each class of problems, we develop bounds on the adaptive optimum, the adaptivity gap, and we develop polynomial-time algorithms approximating the adaptive optimum. In addition, we estimate the respective integrality gaps, randomness gaps and we address the issue of polynomial-time inapproximability.

7.1 Bounding adaptive policies

We provide a general bound on the expected value achieved by any adaptive policy. First, we ignore item values and we analyze the *mean truncated size* (defined in Section 5.2) of all the items that an adaptive policy attempts to insert. I.e., we count all the items inserted including the first overflowing item. This is a generalization of the bounds for Stochastic Knapsack developed in Section 6.1.

Lemma 18. *For a Stochastic Packing problem and any adaptive policy, let $\mathcal{A}$ denote the (random) set of items which the policy attempts to insert. Then for each component j,*

$$\mathbf{E}[\mu_j(\mathcal{A})] \leq b_j + m_j$$

where m_j is the maximum possible truncated size $\min\{S_j(i), b_j\}$ for any item.

Proof. Consider component j. Denote by $M_j(c)$ the maximum possible $\mathbf{E}[\mu_j(\mathcal{A})]$ for a random set $\mathcal{A}$ that an adaptive policy can attempt to insert within capacity c in the j-th component. For now, all other components can be ignored. We prove, by induction on the number of available items, that $M_j(c) \leq c + m_j$.

Suppose that an optimal adaptive policy, given remaining space c, inserts item i. Denote by $fit(i, c)$ the characteristic function of the event that item i fits ($S_j(i) \leq c$) and by $\tilde{s}(i)$ its truncated size $\tilde{s}(i) = \min\{S_j(i), b_j\}$. Having inserted item i, the policy continues only if item i fits and then the remaining space is $c - \tilde{s}(i) \geq 0$:

$$M_j(c) = \mu_j(i) + \mathbf{E}[fit(i, c)M_j(c - \tilde{s}(i))] = \mathbf{E}[\tilde{s}(i) + fit(i, c)M_j(c - \tilde{s}(i))].$$

Applying the induction hypothesis to $M_j(c - \tilde{s}(i))$,

$$\begin{aligned} M_j(c) &\leq \mathbf{E}[\tilde{s}(i) + fit(i,c)(c - \tilde{s}(i) + m_j)] \\ &= \mathbf{E}[fit(i,c)(c + m_j) + (1 - fit(i,c))\ \tilde{s}(i)] \\ &\leq \mathbf{E}[fit(i,c)(c + m_j) + (1 - fit(i,c))\ m_j] \\ &\leq c + m_j. \end{aligned}$$

□

We will write a linear program bounding the expected value obtained by any adaptive policy. Again, we consider the total probability x_i that the adaptive policy attempts to insert item i. In case of a problem with item multiplicity, let x_i denote the expected number of inserted items of type i. Also, for each item define an *effective value* which is an upper bound on the expected profit a policy can gain if it attempts to insert the item.

Definition 14. *For an item i and capacity $\vec{b}$, the effective value is*

$$w_i = v_i\ \Pr[\vec{S}(i) \leq \vec{b}].$$

Theorem 19. *For an instance of Stochastic Packing, $ADAPT \leq \Phi(\vec{b} + \vec{m})$ where*

$$\Phi(\vec{c}) = \max \left\{ \sum_i x_i w_i : \begin{array}{l} \sum_i x_i \vec{\mu}(i) \leq \vec{c} \\ \forall i; x_i \in [0,1] \end{array} \right\}.$$

For an instance of Stochastic Packing with multiplicity, $ADAPT \leq \Phi^+(\vec{b} + \vec{m})$ where

$$\Phi^+(\vec{c}) = \max \left\{ \sum_i x_i w_i : \begin{array}{l} \sum_i x_i \vec{\mu}(i) \leq \vec{c} \\ \forall i; x_i \geq 0 \end{array} \right\}.$$

Proof. The expected value achieved by a policy is at most $\sum_i x_i w_i$, where x_i is the probability that the policy attempts to insert item i (or the expected number of copies inserted). The expected mean size vector is $\mathbf{E}[\vec{\mu}(\mathcal{A})] = \sum_i x_i \vec{\mu}(i)$. We know that this is bounded by $\mathbf{E}[\vec{\mu}(\mathcal{A})] \leq \vec{b} + \vec{m}$ where $\vec{b}$ is the capacity and m_j is the maximum possible truncated size of any item in component j. Thus $\vec{x}$ is a feasible vector for the LP and $\sum_i x_i w_i$ is bounded by the LP optimum. □

Note the similarity between these LPs and the usual linear relaxations of packing problems. We just replace random sizes $\vec{S}(i)$ by mean sizes $\vec{\mu}(i)$, item values v_i by effective values w_i and we modify the right-hand side. These LPs will be used in the following sections to design and analyze non-adaptive policies.

7.2 Stochastic Packing and hardness of PIP

Consider the most general variant of Stochastic Packing where item sizes are unrestricted vectors in $\mathbb{R}^d_+$. We shall see that a straightforward generalization of the greedy randomized algorithm from Section 6.2 gives an $O(d)$-approximation.

Since we are dealing with general item sizes, we can assume by scaling that $\vec{b} = (1, 1, \ldots, 1)$. It will be convenient to use the l_1 norm of the mean size vector as a measure of multidimensional size:

$$||\vec{\mu}(\mathcal{A})||_1 = \sum_{j=1}^{d} \mu_j(\mathcal{A}).$$

The reason to use the l_1 norm here is that it bounds the probability that a set of items is a feasible solution. Also, the l_1 norm is easy to work with, because it's additive for collections of items.

Lemma 19.

$$\Pr[||\vec{S}(\mathcal{A})||_\infty \le 1] \ge 1 - ||\vec{\mu}(\mathcal{A})||_1.$$

Proof. For each component, $\Pr[S_j(\mathcal{A}) \ge 1] \le \mathbf{E}[\min\{S_j(\mathcal{A}), 1\}] \le \mu_j(\mathcal{A})$, and by the union bound $\Pr[||\vec{S}(\mathcal{A})||_\infty \ge 1] \le \sum_{j=1}^{d} \mu_j(\mathcal{A}) = ||\vec{\mu}(\mathcal{A})||_1$. □

We design an algorithm for Stochastic Packing analogous to the randomized greedy algorithm for Stochastic Knapsack. Let the items be ordered by value density, now with respect to the l_1 norm of mean size:

$$\frac{w_1}{||\vec{\mu}(1)||_1} \ge \frac{w_2}{||\vec{\mu}(2)||_1} \ge \frac{w_3}{||\vec{\mu}(3)||_1} \ge \cdots.$$

As a further relaxation of the LP from Theorem 19, we can write

$$\phi(t) = \max \left\{ \sum_i w_i x_i : \begin{array}{l} \sum_i x_i ||\vec{\mu}(i)||_1 \le t \\ x_i \in [0,1] \end{array} \right\}.$$

Theorem 19 implies that $ADAPT \le \Phi(\vec{2})$ where $\vec{2} = (2, 2, \ldots, 2)$, and therefore $ADAPT \le \phi(2d)$. We can apply the randomized greedy algorithm for Stochastic Knapsack, using the l_1 norm for mean item sizes.

The randomized greedy algorithm. Let r be the minimum such that $M_r = \sum_{i=1}^{r} ||\vec{\mu}(i)||_1 \ge 1$. Let $p' = (1 - M_r)/||\vec{\mu}(r)||_1$ and $w'_r = p' w_r$. Let $w'_j = w_j$ for $j < r$ and assume $\phi(1) = \sum_{i=1}^{r} w'_r = 1$.

- Choose $k \in \{1, 2, \ldots, r\}$ with probability w'_k.
- If $k < r$, insert item k first. If $k = r$, flip another independent coin $Be(p')$ and insert item r in case of success.
- Insert items $1, 2, \ldots, k-1, k+1, \ldots$ in the greedy order.

We refer to the analysis in Section 6.2. Thanks to Lemma 19, all estimates on the performance of our algorithm are valid upon replacing μ by $||\vec{\mu}||_1$. The adaptive optimum is bounded by $\phi(2d) \leq d\phi(2)$ which yields an additional factor of d.

Theorem 20. *For any Stochastic Packing problem in dimension d,*

$$ADAPT \leq \frac{32d}{7}\ NONADAPT$$

and the corresponding non-adaptive policy can be found in polynomial time.

In particular, our randomized greedy algorithm also provides a $(32d/7)$-approximation algorithm for PIP. Of course, an $O(d)$ -approximation for PIP is known [39] and quite easy to achieve. An approximation factor of $O(d)$ may seem rather weak but in fact it cannot be improved significantly. It is unlikely that a $d^{1-\epsilon}$-approximation can be achieved in polynomial time. We prove this result by tweaking a known reduction which shows $d^{1/2-\epsilon}$-inapproximability for set packing [10].

Theorem 21. *There is no polynomial-time $d^{1-\epsilon}$-approximation algorithm for PIP for any $\epsilon > 0$, unless $NP = ZPP$.*

Proof. We use Håstad's result on the inapproximability of Max Clique [46], or more conveniently maximum independent set. For a graph G, we define the following PIP instance.

Let $A \in \mathbb{R}_+^{d \times n}$ be a matrix where $d = n = |V(G)|$, $A(i,i) = 1$ for every i, $A(i,j) = 1/n$ for $(i,j) \in E(G)$ and $A(i,j) = 0$ otherwise. Let $\vec{b} = \vec{v} = (1, 1, \ldots 1)$. It is easy to see that $A\vec{x} \leq \vec{b}$ for $x \in \{0,1\}^n$ if and only if $\vec{x}$ is the characteristic vector of a stable set. Therefore approximating the optimum of this PIP to within $d^{1-\epsilon}$ for any $\epsilon > 0$ would imply a $n^{1-\epsilon}$-approximation algorithm for maximum stable set, which would imply $NP = ZPP$. □

7.3 The benefit of adaptivity

The greedy algorithm implies that the adaptivity gap for Stochastic Packing is always $O(d)$. We do not know whether this is tight. The best lower bound that we have is $\Omega(\sqrt{d})$. The example is a simple instance of Set Packing.

Lemma 20. *There are instances of Stochastic Set Packing such that*

$$ADAPT \geq \frac{\sqrt{d}}{2}\ NONADAPT.$$

Proof. Define items of type $i = 1, 2, \ldots, d$, where items of type i have size vector $\vec{S}(i) = Be(p)\ \vec{e}_i$, i.e. a random Bernoulli variable in the i-th component, and 0 in the remaining components ($p > 0$ to be chosen later). We have an unlimited supply of items of each type. All items have unit value and we assume unit capacity ($\vec{b} = (1, 1, \ldots 1)$).

An adaptive policy can insert items of each type until a size of 1 is attained in the respective component; the expected number of items of each type inserted is $1/p$. Therefore $ADAPT \geq d/p$.

On the other hand, consider a set of items $\mathcal{F}$. We estimate the probability that F is a feasible solution. For every component i, let k_i denote the number of items of type i in $\mathcal{F}$. We have:

$$\Pr[S_i(\mathcal{F}) \leq 1] = (1-p)^{k_i} + k_i p(1-p)^{k_i-1} =$$

$$(1+p(k_i-1))(1-p)^{k_i-1} \leq (1+p)^{k_i-1}(1-p)^{k_i-1},$$

and

$$\Pr[||\vec{S}(\mathcal{F})||_\infty \leq 1] \leq \prod_{i=1}^{d} (1-p^2)^{k_i-1} \leq e^{-p^2 \sum (k_i-1)} = e^{-p^2(|\mathcal{F}|-d)}.$$

Thus the probability that a set of items fits decreases exponentially with the number of items. For any non-adaptive policy, the probability that the first k items in the sequence are inserted successfully is at most $e^{-p^2(k-d)}$, and we can estimate the expected value achieved :

$$NONADAPT = \sum_{k=1}^{\infty} \Pr[k \text{ items fit}] \leq d + \sum_{k=d+1}^{\infty} e^{-p^2(k-d)} = d + \frac{1}{e^{p^2}-1} \leq d + \frac{1}{p^2}.$$

We choose $p = 1/\sqrt{d}$ which yields $ADAPT \geq d^{3/2}$ and $NONADAPT \leq 2d$. □

We generalize this example to an arbitrary integer capacity vector $\vec{b}$.

Lemma 21. *There are instances of Stochastic Packing with $A \in \{0,1\}^{d \times n}$ and $\vec{b} = (b_1, b_2, \ldots b_d) \in \mathbb{Z}_+^d$, such that*

$$ADAPT \geq \frac{\lambda}{4}\ NONADAPT$$

where $\lambda \geq 1$ is the solution of $\sum_{i=1}^{d} 1/\lambda^{b_i+1} = 1$.

Proof. Let $p \leq 1$ satisfy $\sum_{i=1}^{d} p^{b_i+1} = 1$. Consider the same set of items that we used in the previous proof, only the values are modified as $v_i = p^{b_i+1}/(b_i+1)$. The same adaptive strategy will now insert items of each type, until it accumulates size b_i in the i-th component. The expected number of items of type i inserted will be b_i/p, and therefore

$$ADAPT \geq \sum_{i=1}^{d} v_i \frac{b_i}{p} = \frac{1}{p} \sum_{i=1}^{d} \frac{b_i p^{b_i+1}}{b_i+1} \geq \frac{1}{2p}.$$

Consider a set of items $\mathcal{F}$. We divide the items of each type into blocks of size b_i+1 (for type i). Suppose that the number of blocks of type i is k_i. We estimate the probability that $\mathcal{F}$ is a feasible solution; we use the fact that each block alone has a

probability of overflow p^{b_i+1}, and these events are independent:

$$\Pr[S_i(\mathcal{F}) \leq b_i] \leq (1-p^{b_i+1})^{k_i} \leq e^{-k_i p^{b_i+1}},$$

$$\Pr[\vec{S}(\mathcal{F}) \leq \vec{b}] \leq e^{-\sum k_i p^{b_i+1}}.$$

Now we express this probability as a function of the value of $\mathcal{F}$. We defined the blocks in such a way that a block of type i gets value p^{b_i+1}, and $\sum k_i p^{b_i+1}$ is the value of all the blocks. There might be items of value less than p^{b_i+1} of type i, which are not assigned to any block. All these together can have value at most 1 (by the definition of p). Therefore for any set of value $v(\mathcal{F}) = 1+w$,

$$\Pr[\vec{S}(\mathcal{F}) \leq \vec{b}] \leq e^{-w}.$$

Thus we can estimate the expected value achieved by any non-adaptive policy:

$$NONADAPT \leq 1 + \int_0^\infty \Pr[\text{set of value } \geq 1+w \text{ fits}]dw \;\leq\; 1 + \int_0^\infty e^{-w}dw = 2.$$

It follows that the adaptivity gap is at least $1/4p = \lambda/4$, where λ satisfies $\sum_{i=1}^{d} 1/\lambda^{b_i+1} = 1$. □

As a special case, for $\vec{b} = (B, B, \ldots, B)$, the lemma holds with $\lambda = d^{1/(B+1)}$. Note how the adaptivity gaps for Stochastic Set Packing and b-matching follow closely the known approximability factors for Set Packing and b-matching (see Section 5.4). In Section 7.4, we prove that for Stochastic Set Packing and b-matching, these bounds on the adaptivity gap are not only tight, but they can be actually achieved by a polynomial-time non-adaptive policy.

7.4 Stochastic Set Packing and b-matching

Consider the special case of Set Packing: we assume that the random size vectors have values in $\{0,1\}^d$ and the capacity vector is $\vec{b} = (1, 1, \ldots 1)$. In other words, each item is given by a probability distribution over subsets of $[d]$ and we want to pack disjoint sets of maximum value. In this case, the adaptivity gap can be $\Omega(\sqrt{d})$ (see Section 7.3) while our greedy algorithm provides only an $O(d)$-approximation. Here, we improve this to $O(\sqrt{d})$ and thus close the gap up to a constant factor.

Our solution is a fixed collection of items - that is, we insert all these items, and we collect nonzero profit only if all the respective sets turn out to be disjoint. The first step is to replace the l_1 norm by a stronger measure of size, which allows one to estimate better the probability that a collection of items is a feasible solution. Fortunately, the restriction to $\{0,1\}^d$ vectors makes this possible.

Definition 15. *For a set of items $\mathcal{A}$, define*

$$\hat{\mu}(\mathcal{A}) = \sum_{\{i,j\} \in \binom{\mathcal{A}}{2}} \vec{\mu}(i) \cdot \vec{\mu}(j).$$

Lemma 22. *For a set of items $\mathcal{A}$ with size vectors in $\{0,1\}^d$,*

$$\Pr[||\vec{S}(\mathcal{A})||_\infty \leq 1] \geq 1 - \hat{\mu}(\mathcal{A}).$$

Proof. A set of items can overflow in coordinate l, only if at least two items attain size 1 in that coordinate. For a pair of items $\{i,j\}$, the probability of this happening is $\mu_l(i)\mu_l(j)$. By the union bound:

$$Pr[S_l(\mathcal{A}) > 1] \leq \sum_{\{i,j\} \in \binom{\mathcal{A}}{2}} \mu_l(i)\mu_l(j),$$

$$Pr[||\vec{S}(\mathcal{A})||_\infty > 1] \leq \sum_{\{i,j\} \in \binom{\mathcal{A}}{2}} \vec{\mu}(i) \cdot \vec{\mu}(j) = \hat{\mu}(\mathcal{A}).$$

□

Theorem 22. *For Stochastic Set Packing,*

$$ADAPT \leq 5.6\sqrt{d}\ NONADAPT$$

and the corresponding non-adaptive policy can be found in polynomial time.

Proof. We use the LP formulation introduced in Section 7.1. Since we can solve the LP in polynomial time, we can assume that we have a solution x such that $||\sum x_i \vec{\mu}(i)||_\infty \leq 2$ and $\mathcal{V} = \sum x_i v_i$ bounds the expected value of any adaptive policy (Theorem 19; in this case, $w_i = v_i$). We can also assume that the value of any item is at most $\frac{\beta}{\sqrt{d}}\mathcal{V}$ for some fixed $\beta > 0$, otherwise the most valuable item alone is a $\frac{\sqrt{d}}{\beta}$-approximation of the optimum.

We sample a random set of items $\mathcal{F}$, item i with probability $q_i = \frac{\alpha}{\sqrt{d}} x_i$. Constants α, β will be chosen later. We estimate the expected value that we get for the set obtained in this way. Note that there are "two levels of expectation" here: one related to our sampling, and another to the resulting set being used as a solution of a stochastic problem. The expectation denoted by $\mathbf{E}[\ldots]$ in the following computation is the one related to our sampling. Using Lemma 22, we can lower bound the expected value achieved by inserting set $\mathcal{F}$.

$$\mathbf{E}[v(\mathcal{F})(1 - \hat{\mu}(\mathcal{F}))] = \mathbf{E}\left[\sum_{i \in \mathcal{F}} v_i - \sum_{i \in \mathcal{F}} v_i \sum_{\{j,k\} \in \binom{\mathcal{F}}{2}} \vec{\mu}(j) \cdot \vec{\mu}(k)\right]$$

and by multiplying out and separating the terms where $i = j$, $i = k$ and $i \notin \{j,k\}$,

we get

$$\begin{aligned}\mathbf{E}[v(\mathcal{F})(1-\hat{\mu}(\mathcal{F}))] &= \textstyle\sum_i q_i v_i - \sum_{j,k} q_j q_k v_j \vec{\mu}(j)\cdot\vec{\mu}(k) - \frac{1}{2}\sum_{i,j,k} q_i q_j q_k v_i \vec{\mu}(j)\cdot\vec{\mu}(k)\\ &\geq \tfrac{\alpha}{\sqrt{d}}\textstyle\sum_i x_i v_i - \frac{\alpha^2\beta}{d^{3/2}}\mathcal{V}\sum_{j,k} x_j x_k \vec{\mu}(j)\cdot\vec{\mu}(k) - \frac{\alpha^3}{2d^{3/2}}\sum_i x_i v_i \sum_{j,k} x_j x_k \vec{\mu}(j)\cdot\vec{\mu}(k)\\ &\geq \tfrac{\alpha}{\sqrt{d}}\mathcal{V} - \tfrac{\alpha^2\beta}{d^{3/2}}\mathcal{V}||\textstyle\sum_j x_j\vec{\mu}(j)||^2 - \frac{\alpha^3}{2d^{3/2}}\mathcal{V}||\sum_j x_j\vec{\mu}(j)||^2\\ &\geq \tfrac{\alpha}{\sqrt{d}}(1-4\alpha\beta-2\alpha^2)\mathcal{V}\end{aligned}$$

where we have used $v_j \leq \frac{\beta}{\sqrt{d}}\mathcal{V}$.

We choose α and β to satisfy $\alpha(1-4\alpha\beta-2\alpha^2)=\beta$ and then maximize this value, which yields $\alpha^2 = (-5+\sqrt{33})/8$ and $\beta^2 = (11\sqrt{33}-59)/128$. Then $\sqrt{d}/\beta < 5.6\sqrt{d}$ is our approximation factor. This closes the gap for the Stochastic Set Packing problem up to a constant factor because we know from Section 7.3 that the adaptivity gap can be as large as $\frac{1}{2}\sqrt{d}$. Finally, using the method of conditional expectations (on $\mathbf{E}[v(\mathcal{F})(1-\hat{\mu}(\mathcal{F}))]$ which can be computed exactly), we can find a good set $\mathcal{F}$ deterministically. □

Next we show how this algorithm generalizes to b-matching, with an arbitrary integer vector $\vec{b}$. A natural generalization of $\hat{\mu}(\mathcal{A})$ and Lemma 22 is the following.

Definition 16. *For a set of items $\mathcal{A}$,*

$$\hat{\mu}_{\vec{b}}(\mathcal{A}) = \sum_{l=1}^{d} \sum_{B\in\binom{\mathcal{A}}{b_l+1}} \prod_{i\in B} \mu_l(i).$$

Lemma 23. *For a set of items $\mathcal{A}$ with size vectors in $\{0,1\}^d$,*

$$\Pr[\vec{S}(\mathcal{A}) \leq \vec{b}] \geq 1 - \hat{\mu}_{\vec{b}}(\mathcal{A}).$$

Proof. Similarly to Lemma 22, a set of items can overflow in coordinate l, only if b_l+1 items attain size 1 in their j-th component. This happens with probability $\prod_{i\in B}\mu_l(i)$ and we apply the union bound. □

Theorem 23. *For Stochastic b-matching,*

$$ADAPT \leq 20.3\lambda\ NONADAPT$$

where $\lambda \geq 1$ is the solution of $\sum_j 1/\lambda^{b_j+1} = 1$. The corresponding non-adaptive policy can be found in polynomial time for any fixed capacity $\vec{b}$.

Proof. We solve

$$\mathcal{V} = \max\left\{ \sum_i x_i v_i : \begin{array}{ll} \sum_i x_i \mu_l(i) \leq b_l + 1 & \forall l \\ 0 \leq x_i \leq 1 & \forall i \end{array} \right\}$$

which is an upper bound on the adaptive optimum. We assume that the value of each item is at most $\frac{\beta}{\lambda}\mathcal{V}$ and we sample $\mathcal{F}$, each item i with probability $q_i = \frac{\alpha}{\lambda}x_i$, where $\sum_j 1/\lambda^{b_j+1} = 1$; $\alpha, \beta > 0$ to be chosen later. We estimate the expected value of $\mathcal{F}$ in a way similar to the Set Packing case.

$$\begin{aligned}
&\mathbf{E}[v(\mathcal{F})(1-\mu_{\vec{b}}(\mathcal{F}))] \\
&= \mathbf{E}\left[\sum_{i\in\mathcal{F}} v_i - \sum_{l=1}^{d}\sum_{B\in\binom{\mathcal{F}}{b_l+1}}\sum_{i\in B} v_i \prod_{j\in B}\mu_l(j) - \sum_{l=1}^{d}\sum_{B\in\binom{\mathcal{F}}{b_l+1}}\sum_{i\in\mathcal{F}\setminus B} v_i\prod_{j\in B}\mu_l(j)\right] \\
&\geq \sum_i q_i v_i - \frac{\beta}{\lambda}\mathcal{V}\sum_{l=1}^{d}(b_l+1)\sum_{|B|=b_l+1}\prod_{j\in B} q_j\mu_l(j) - \sum_{l=1}^{d}\sum_i q_i v_i \sum_{|B|=b_l+1}\prod_{j\in B} q_j\mu_l(j) \\
&\geq \frac{\alpha}{\lambda}\mathcal{V} - \frac{\beta}{\lambda}\mathcal{V}\sum_{l=1}^{d}\frac{1}{b_l!}\left(\sum_i q_i\mu_l(i)\right)^{b_l+1} - \frac{\alpha}{\lambda}\mathcal{V}\sum_{l=1}^{d}\frac{1}{(b_l+1)!}\left(\sum_i q_i\mu_l(i)\right)^{b_l+1} \\
&\geq \frac{\alpha}{\lambda}\mathcal{V} - \frac{\beta}{\lambda}\mathcal{V}\sum_{l=1}^{d}\frac{1}{b_l!}\left(\frac{\alpha}{\lambda}(b_l+1)\right)^{b_l+1} - \frac{\alpha}{\lambda}\mathcal{V}\sum_{l=1}^{d}\frac{1}{(b_l+1)!}\left(\frac{\alpha}{\lambda}(b_l+1)\right)^{b_l+1}.
\end{aligned}$$

We use Stirling's formula, $(b_l+1)! > \left(\frac{b_l+1}{e}\right)^{b_l+1}$, and $b_l! > \frac{(b_l+1)!}{2^{b_l+1}} > \left(\frac{b_l+1}{2e}\right)^{b_l+1}$. Also, we assume $2e\alpha < 1$.

$$\begin{aligned}
&\mathbf{E}[v(\mathcal{F})(1-\mu_{\vec{b}}(\mathcal{F}))] \\
&\geq \frac{\alpha}{\lambda}\mathcal{V} - \frac{\beta}{\lambda}\mathcal{V}\sum_{l=1}^{d}\left(\frac{2e\alpha}{\lambda}\right)^{b_l+1} - \frac{\alpha}{\lambda}\mathcal{V}\sum_{l=1}^{d}\left(\frac{e\alpha}{\lambda}\right)^{b_l+1} \\
&\geq \frac{\alpha}{\lambda}\mathcal{V} - \frac{2e\alpha\beta}{\lambda}\mathcal{V}\sum_{l=1}^{d}\frac{1}{\lambda^{b_l+1}} - \frac{e\alpha^2}{\lambda}\mathcal{V}\sum_{l=1}^{d}\frac{1}{\lambda^{b_l+1}} = \frac{\alpha}{\lambda}\mathcal{V}(1-2e\beta-e\alpha).
\end{aligned}$$

We choose optimally $2e\alpha = -1+\sqrt{3}$ and $2e\beta = 2-\sqrt{3}$ which gives an approximation factor of $2e\lambda/(2-\sqrt{3}) < 20.3\lambda$. For constant $\vec{b}$, we can compute conditional expectations in polynomial time and thereby derandomize the algorithm. □

This result is tight up to a constant factor with respect to the adaptivity gap for b-matching. In the deterministic case, for $\vec{b} = (B, B, \ldots B)$, there is no polynomial-time $d^{1/(B+1)-\epsilon}$-approximation unless $ZPP = NP$ [10]. Thus our approximation algorithm is essentially optimal even in the deterministic case.

7.5 Restricted Stochastic Packing

As the last variant of Stochastic Packing, we consider instances where the item sizes are general vectors restricted to $\vec{S}(i) \in [0,1]^d$ and the capacity vector is a given vector $\vec{b} \in \mathbb{R}_+^d$. Similarly to b-matching, we prove an approximation factor as a function of

the parameter $\vec{b}$, and we find that our approach is particularly successful in case of capacity $\vec{b}$ very large compared to item sizes.

Theorem 24. *For Stochastic Packing with item sizes $\vec{S}(i) \in [0,1]^d$ and capacity $\vec{b}$,*

$$ADAPT \leq 120\lambda \; NONADAPT$$

where $\lambda \geq 1$ is the solution of $\sum_{i=1}^{d} 1/\lambda^{b_i} = 1$. The corresponding non-adaptive policy can be found in polynomial time.

Proof. Consider the LP bounding the performance of any adaptive policy:

$$\mathcal{V} = \max \left\{ \sum_i x_i v_i : \; \sum_i x_i \vec{\mu}(i) \leq \vec{b} + \vec{1}, \; x_i \in [0,1] \right\}.$$

Assume that $v_i \leq \frac{\beta}{\lambda}\mathcal{V}$ for each item i, for some constant $\beta > 0$ to be chosen later, otherwise one item alone is a good approximation solution. We find an optimal solution x and define

$$q_i = \frac{\alpha}{\lambda} x_i,$$

$\alpha > 0$ again to be chosen later.

Our randomized non-adaptive policy inserts item i with probability q_i. Let's estimate the probability that this random set of items $\mathcal{F}$ fits, with respect to both (independent) levels of randomization - our randomized policy and the random item sizes. For each coordinate j,

$$\mathbf{E}[S_j(\mathcal{F})] = \sum_i q_i \mu_j(i) \leq \frac{\alpha}{\lambda}(b_j + 1).$$

Since this is a sum of $[0,1]$ independent random variables, we apply the Chernoff bound to estimate the probability of overflow (use $\mu \leq \alpha(b_j+1)/\lambda, 1+\delta = b_j/\mu$):

$$\Pr[S_j(\mathcal{F}) > b_j] < \left(\frac{e^\delta}{(1+\delta)^{1+\delta}} \right)^\mu < \left(\frac{e}{1+\delta} \right)^{(1+\delta)\mu} \leq \left(\frac{e\mu}{b_j} \right)^{b_j} \leq \left(\frac{2e\alpha}{\lambda} \right)^{b_j}.$$

Using the union bound,

$$\Pr[\exists j; S_j(\mathcal{F}) > b_j] < 2e\alpha \sum_{j=1}^{d} \frac{1}{\lambda^{b_j}} = 2e\alpha.$$

Now we estimate the probability that the value of $\mathcal{F}$ is too low. We assume that $v_i \leq \frac{\beta}{\lambda}\mathcal{V}$, therefore by scaling we obtain values $w_i = \frac{\lambda}{\beta\mathcal{V}} v_i \in [0,1]$. We sample each of them with probability q_i which yields a random sum W with expectation $\mathbf{E}[W] = \sum_i q_i w_i = \alpha/\beta$. Again by Chernoff bound,

$$\Pr\left[W < \frac{1}{2}\mathbf{E}[W] \right] < e^{-\mathbf{E}[W]/8} = e^{-\alpha/8\beta}.$$

We choose $\alpha = 1/10$ and $\beta = 1/100$ which yields $\Pr[\exists j; S_j(\mathcal{F}) > b_j] < 2e\alpha < 0.544$ and $\Pr[v(\mathcal{F}) < \frac{1}{20\lambda}\mathcal{V}] < e^{-\alpha/8\beta} < 0.287$, which means that with probability at least 0.169, we get a feasible solution of value $\frac{1}{20\lambda}\mathcal{V}$. In any case, the expected value achieved by our randomized policy is at least $\frac{1}{120\lambda}\mathcal{V}$.

Finally, note that any randomized non-adaptive policy can be seen as a convex linear combination of deterministic non-adaptive policies. Therefore there is also a deterministic non-adaptive policy achieving at least the expectation. A fixed set achieving expected value at least $ADAPT/120\lambda$ can be found using the method of pessimistic estimators applied to the Chernoff bounds. This is analogous to the derandomization for Packing Integer Programs developed by Raghavan [39]. □

As we mentioned, the best approximation in the deterministic case is known to be $O(d^{1/B})$ where $B = \min_i b_i$ (see [39]; in this case, our algorithm is very similar to Raghavan's). The hardness result from [10] says that for integer B, it is hard to approximate restricted PIP to within $d^{1/(B+1)-\epsilon}$. We strengthen this result to $d^{1/B-\epsilon}$, to show that that the randomized rounding approach is essentially optimal.

Theorem 25. *There is no polynomial-time $d^{1/B-\epsilon}$-approximation algorithm for PIP with $A \in [0,1]^{d\times n}$ and $\vec{b} = (B, B, \ldots, B)$, $B \in \mathbb{Z}, B \geq 2$, unless $NP = ZPP$.*

Proof. For a given graph $G = (V, E)$, denote by d the number of B-cliques ($d < n^B$). Define a $d \times n$ matrix A (i.e., indexed by the B-cliques and vertices of G), where $A_{Q,v} = 1$ if vertex v belongs to clique Q, $A_{Q,v} = 1/n$ if vertex v is connected by an edge to clique Q, and $A_{Q,v} = 0$ otherwise. Let the value vector be $\vec{v} = (1, 1, \ldots, 1)$ and denote the optimum of this PIP by $\mathcal{V}$. Let $\epsilon > 0$ be arbitrarily small, and assume that we can approximate $\mathcal{V}$ to within a factor of $d^{1/B-\epsilon}$.

Suppose that $\vec{x}$ is the characteristic vector of an independent set $S, |S| = \alpha(G)$. Then $A\vec{x} \leq \vec{b}$ because in any clique, there is at most one member of S and the remaining vertices contribute at most $1/n$ each. Thus the optimum of the PIP is $\mathcal{V} \geq \sum_v x_v = \alpha(G)$.

If $A\vec{x} \leq \vec{b}$ for some $\vec{x} \in \{0,1\}^n$, then the subgraph induced by $S = \{v : x_v = 1\}$ cannot have a clique larger than B: suppose $R \subseteq S$ is a clique of size $B + 1$, and $Q \subset R$ a sub-clique of size B. Then $(A\vec{x})_Q = \sum_v A_{Q,v}x_v > B$, since it collects 1 from each vertex in Q plus at least $1/n$ from the remaining vertex in $R \setminus Q$. Finally, we invoke a lemma from [10] which claims that a subgraph on $|S| = \sum_v x_v$ vertices without cliques larger than B must have an independent set of size $\alpha(G) \geq |S|^{1/B}$, i.e. $\mathcal{V} \leq (\alpha(G))^B$.

We assume that we can find $\mathcal{W}$ such that $\mathcal{V}/d^{1/B-\epsilon} \leq \mathcal{W} \leq \mathcal{V}$ and we prove that this would imply an $n^{1-\epsilon}$-approximation to $\alpha(G)$: We report $a = \mathcal{W}^{1/B}$ as our answer to approximate $\alpha(G)$. On the one hand, we know that $a \leq \mathcal{V}^{1/B} \leq \alpha(G)$. On the other hand,

$$a \geq \left(\frac{\mathcal{V}}{d^{1/B-\epsilon}}\right)^{1/B} \geq \frac{(\alpha(G))^{1/B}}{n^{1/B-\epsilon}} \geq \frac{\alpha(G)}{n^{1-\epsilon}}$$

where we have used $\mathcal{V} \geq \alpha(G)$, $d < n^B$, and finally $\alpha(G) \leq n$. This proves that a is an $n^{1-\epsilon}$-approximation to $\alpha(G)$, which would imply $NP = ZPP$. □

7.6 Integrality and randomness gaps

As we have seen, the adaptivity gaps in general dimension d grow as functions of d, depending on the specific class of problems, and they seem to have some underlying connection with the approximation factors that can be achieved algorithmically. Another quantity that's often related to deterministic approximability is the integrality gap, and another one that we defined in Section 5.3 is the randomness gap. All these factors exhibit very similar behavior as functions of d. It appears that more structure is hidden under the surface than we could unravel in this thesis.

One connection, which is partially clarified here, is between the randomness and integrality gaps (defined in Sections 5.2 and 5.3) for packing and covering problems with multiplicity. We prove that the randomness gap is related to the integrality gap of a linear program associated with the stochastic optimization problem, as stated in the following theorem.

Theorem 26. *If a PIP instance, with item multiplicity, each item size in a domain $\mathcal{D}, \{0,1\}^d \subseteq \mathcal{D} \subseteq [0,1]^d$, unit item values, and capacity $\vec{b} = (B, B, \ldots, B), B \in \mathbb{Z}_+$, has integrality gap ω, then there is a pair of Stochastic Packing instances with multiplicity, one item type with random size in $\mathcal{D}$, and capacity $\vec{b}$, such that the randomness gap for the two instances is $\rho_a = \rho_n \geq \omega$.*

Proof. Consider such a PIP instance and its corresponding linear relaxation

$$LP = \max \left\{ \sum_i x_i : \sum_i x_i \vec{a}(i) \leq \vec{b},\ x_i \geq 0 \right\} \tag{7.1}$$

with integrality gap ω. In other words, at most LP/ω items can fit integrally.

Let x_i be an optimal solution, i.e. $\sum x_i = LP$ and $\sum x_i \vec{a}(i) \leq \vec{b}$. We define two Stochastic Packing instances, each with only one item type. In the first one, with probability $p_i = x_i/LP$, let an item have size $\vec{S} = \vec{a}(i)$. The expected size is

$$\vec{\mu} = \mathbf{E}[\vec{S}] = \sum_i \frac{x_i}{LP} \vec{a}(i) \leq \frac{1}{LP} \vec{b}. \tag{7.2}$$

However, whatever the instantiated sizes $\vec{a}(i)$ in a sequence of items, we know that at most LP/ω items can fit.

In the second instance, we define an item type of random size $\vec{T}$ with values in $\{0,1\}^d \subseteq \mathcal{D}$: we generate a uniformly random number $\xi \in [0,1]$; for each j, if $\xi \leq \mu_j$, we set $T_j = 1$, otherwise $T_j = 0$. Thus $\mathbf{E}[\vec{T}] = \vec{\mu} = \mathbf{E}[\vec{S}]$. However, the components of $\vec{T}$ are positively correlated in such a way that if $T_j = 0$ for the component maximizing μ_j then all the remaining components are zero as well. Due to (7.2), $\max \mu_j \leq B/LP$; i.e., each item has non-zero size with probability at most B/LP. We can insert a sequence of these items until one of them attains non-zero size, which will take an expected number of at least LP/B trials. Since the size vectors are contained in $\mathcal{D} \subseteq [0,1]^d$, we can iterate this B times, for a total expected value at least LP.

The same result applies to the non-adaptive randomness gaps since in our instances, there is only one item type and therefore no distinction between adaptive and non-adaptive policies. □

Integrality gaps are known for many combinatorial problems, including PIP and Set Packing [38]. We summarize these results, with extensions to b-matching and Restricted Packing. Finally, from Theorem 26, we get the respective randomness gaps as well.

Lemma 24. *The worst case integrality gaps for PIP are:*

- $\Theta(d)$ *for general PIP.*
- $\Theta(\sqrt{d})$ *for Set Packing.*
- $\Theta(d^{1/(B+1)})$ *for* $A \in \{0,1\}^{d\times n}, \vec{b} = (B,\ldots,B)$
- $\Theta(d^{1/B})$ *for* $A \in [0,1]^{d\times n}, \vec{b} = (B,\ldots,B)$

For stochastic variants of these problems, we get the same randomness gaps.

Proof. Let us demonstrate lower bounds on the integrality gap. Since we shall use only unit item values, then we refer to Theorem 26 which implies the same randomness gaps for the respective Stochastic Packing problems. The upper bounds follow from our approximation algorithms described in the preceding sections.

For general PIP, consider items of size

$$\begin{aligned}\vec{a}(1) &= \left(1, \tfrac{1}{d}, \ldots, \tfrac{1}{d}\right)\\ \vec{a}(2) &= \left(\tfrac{1}{d}, 1, \ldots, \tfrac{1}{d}\right)\\ &\ldots\\ \vec{a}(d) &= \left(\tfrac{1}{d}, \tfrac{1}{d}, \ldots, 1\right)\end{aligned}$$

Fractionally, we can pack $1/2$ of each item which yields a feasible solution of value $d/2$, while integrally only one item fits. So the integrality gap in the general case is at least $d/2$.

For Set Packing, consider a finite projective plane of order $q \geq 2$. (Constructions exist for any prime power q, see [41], Chapter 13, Section 2.) It contains $d = q^2+q+1$ points which form our ground set. It also contains d lines which form our available sets. Through each point, there are $q+1$ lines. Fractionally we can pack each line with a coefficient of $1/(q+1)$ to get a feasible LP solution of value at least $(q^2+q+1)/(q+1) \geq q$. Whereas integrally, we can choose only one line since any two lines intersect. The integrality gap for Set Packing is at least $q \geq \sqrt{d} - 1$.

More generally, consider a finite projective space of order $q \geq 2$ and dimension $B+1$. It contains $d = (q^{B+2}-1)/(q-1)$ points and also d hyperplanes whose characteristic vectors are the item sizes. Since any $B+1$ hyperplanes intersect at a common point, the best solution to the B-matching problem is any choice of B hyperplanes. However, through any point there are only $(q^{B+1}-1)/(q-1)$ hyperplanes, so fractionally we

can pack each hyperplane with a coefficient of $B(q-1)/(q^{B+1}-1)$, for an LP optimum of at least $B(q^{B+2}-1)/(q^{B+1}-1) \geq Bq$. Thus the integrality gap for B-matching is at least $q = \Omega(d^{1/(B+1)})$.

Similarly, define a PIP with $A \in [0,1]^{d \times n}, \vec{b} = (B, \ldots, B)$ corresponding to a finite projective space of order $q \geq 2$ and dimension $B \geq 2$. We have $d = (q^{B+1} - 1)/(q-1)$ points and d hyperplanes which correspond to items. The size vector $\vec{a}(H)$ corresponding to hyperplane H contains 1 in each component $i \in H$ and $1/d$ in each component $i \notin H$. Since any B hyperplanes have a common point of intersection, which yields a size of B in the respective coordinate, and any additional hyperplane would contribute a nonzero amount to this coordinate, we cannot pack more than B items integrally. On the other hand, there are $(q^B - 1)/(q-1)$ hyperplanes through every point and the contribution of all other hyperplanes is at most 1 so we can pack $(B-1)(q-1)/(q^B-1)$ of each hyperplane fractionally, for an LP value of $(B-1)(q^{B+1}-1)/(q^B-1) \geq (B-1)q$. The integrality gap is at least $(1-1/B)q = \Omega(d^{1/B})$. □

Note. These gaps, as well as our adaptivity gaps, apply also to the problem variants with item multiplicity, while our algorithmic approximation results hold with or without multiplicity. In short, item multiplicity does not make a significant difference for Stochastic Packing.

In summary, we have shown that our approximation results are tight up to a constant factor, with respect to integrality and randomness gaps. This implies two facts:

- We cannot obtain better approximation factors for the deterministic packing problems by comparing against the LP optimum.
- We cannot obtain better bounds for stochastic approximation, adaptive or non-adaptive, using only mean item sizes.

Chapter 8

Stochastic Covering

In this chapter, we focus on Stochastic Covering problems, introduced in Chapter 5. The deterministic forefather of these problems is the well known Set Cover problem. In contrast to Stochastic Packing, allowing or not allowing item multiplicity makes a big difference here, and we will distinguish carefully between these two variants.

8.1 Bounding adaptive policies

First we develop linear programs (in analogy to Section 7.1) bounding the performance of adaptive policies. We estimate the mass of items that an adaptive policy needs to insert to achieve a feasible covering, i.e. their *mean truncated size* (see Section 5.2). This bound does not depend on item multiplicity.

Lemma 25. *For a Stochastic Covering problem and any adaptive policy, let $\mathcal{A}$ denote the (random) set of items which the policy uses to achieve a feasible covering. Then for each component j,*

$$\mathbf{E}[\mu_j(\mathcal{A})] \geq b_j.$$

Proof. Consider component j. Denote by $M_j(c)$ the minimum expected $\mu_j(\mathcal{A})$ for a set $\mathcal{A}$ that an adaptive policy inserts to cover capacity c in the j-th component. We prove, by induction on the number of available items, that $M_j(c) \geq c$. Suppose that an optimal adaptive policy, given remaining space c, inserts item i. Denote by $cover(i, c)$ the characteristic function of the event that item i covers the remaining space (i.e., $S_j(i) \geq c$, and in that case the policy terminates). We denote by $\tilde{s}(i)$ the truncated size $\tilde{s}(i) = \min\{S_j(i), b_j\}$:

$$M_j(c) = \mu_j(i) + \mathbf{E}[(1 - cover(i,c))M_j(c - \tilde{s}(i))] = \mathbf{E}[\tilde{s}(i) + (1 - cover(i,c))M_j(c - \tilde{s}(i))]$$

and using the induction hypothesis,

$$M_j(c) \geq \mathbf{E}[\tilde{s}(i) + (1 - cover(i,c))(c - \tilde{s}(i))] = c - \mathbf{E}[cover(i,c)(c - \tilde{s}(i))] \geq c.$$

□

Again, we can write an LP with variables x_i interpreted as the total probability that an adaptive policy inserts item i. We get the following lower bounds on the expected cost of an adaptive solution.

Theorem 27. *For an instance of Stochastic Covering, $ADAPT \geq \Psi(\vec{b})$ where*

$$\Psi(\vec{b}) = \min \left\{ \sum_i x_i v_i : \begin{array}{l} \sum_i x_i \vec{\mu}(i) \geq \vec{b} \\ x_i \in [0,1] \end{array} \right\}.$$

For an instance of Stochastic Covering with multiplicity, $ADAPT \geq \Psi^+(\vec{b})$ where

$$\Psi^+(\vec{b}) = \min \left\{ \sum_i x_i v_i : \begin{array}{l} \sum_i x_i \vec{\mu}(i) \geq \vec{b} \\ x_i \geq 0 \end{array} \right\}.$$

These bounds hold for Stochastic Covering instances with or without multiplicity. However, there are instances without multiplicity where the LP bound is arbitrarily weak.

Example. Consider an instance of Stochastic Set Cover for $d = 1$ and $n = 3$:

- Item 1 has value $v_1 = 1$ and size $S(1) = Be(1/2)$.
- Item 2 has value $v_2 = 1$ and size $S(2) = Be(1/2)$.
- Item 3 has value $v_3 = 1000$ and size $S(3) = 1$.

Note that $x_1 = x_2 = 1, x_3 = 0$ is a feasible solution of the LP and therefore $\Psi(1) \leq 2$. However, any policy will need item 3 with probability at least $1/4$ and so $ADAPT \geq 250$.

Thus the LP bound is not very useful for Stochastic Covering without multiplicity. We will use it only in the case of item multiplicity.

8.2 General Stochastic Covering

Consider the stochastic problem where items of random size $\vec{S}(i)$ and deterministic value v_i are given, and the goal is to achieve a feasible solution such that $\sum_{i \in \mathcal{F}} \vec{S}(i) \geq \vec{b}$. We consider the variant where every item can be used at most once and multiplicity is not allowed. Thus we must assume that there exists a set of items which is feasible with probability 1. We observe quickly that in this setting, there is little we can do.

Lemma 26. *For general Stochastic Covering, the integrality and randomness gaps can be arbitrarily large, even in dimension $d = 1$.*

Proof. Let $\epsilon > 0$ be arbitrarily small. For the integrality gap, consider items of size $S(1) = 1 - \epsilon$ and value $v_1 = 0$, and size $S(2) = 1$ and value $v_2 = 1$. For an integral

solution, we need an item of type 2 for a value of 1, while the optimal fractional solution is $S(1) + \epsilon\ S(2) = 1$ for a value of ϵ.

For the randomness gap, consider 1 item of size $S(1) = 1 - \epsilon$ and value $v_1 = 0$ and an unlimited supply of items of size $S(2) = Be(\epsilon)$ and value $v_2 = 1$. Compare this instance and another one where we replace sizes by their expectations $\mu(1) = 1 - \epsilon$ and $\mu(2) = \epsilon$. With deterministic sizes, $\mu(1) + \mu(2) = 1$ is a feasible solution of cost $ADAPT_2 = 1$. On the other hand, any policy in the stochastic case has to wait for an item of type 2 and size 1, which takes an expected number of $1/\epsilon$ trials. Therefore $ADAPT_1 \geq 1/\epsilon$. □

This indicates that using only mean size vectors, we cannot obtain any approximation algorithm for this problem whatsoever! We do not have a corresponding negative result on the adaptivity gap - conceivably, the adaptivity gap in dimension 1 could be constant. For a lower bound of 7/6, see Section 8.4. However, to prove any kind of upper bound, we would have to resort to a more detailed analysis of item size distributions, with more information available than just the expectation. We leave this question outside the scope of this thesis.

8.3 Stochastic Set Cover

Perhaps the circumstances are more benign in the case of Set Cover, i.e. 0/1 size vectors. There, we know at least that the integrality gap is $O(\log d)$ [32]. However, the worst case randomness gap turns out to be $\Omega(d)$, and we get the same bound for the adaptivity gap as well.

Lemma 27. *For Stochastic Set Cover, the adaptivity gap can be $d/2$ and the randomness gap can be d.*

Proof. For the adaptivity gap, consider $\vec{S}(0) = \vec{1} - \vec{e}_k$, where $k \in \{1, 2, \ldots, d\}$ is uniformly random, $v_0 = 0$, and $\vec{S}(i) = \vec{e}_i$ deterministic, $v_i = 1$, for $i = 1, 2, \ldots, d$. An adaptive policy inserts item 0 first; assume its size is $\vec{S}(0) = \vec{1} - \vec{e}_k$. Then we insert item k which completes the covering for a cost equal to 1. An optimal non-adaptive policy still inserts item 0 first, but then, for any ordering of the remaining items that it chooses, the expected cost incurred before it hits the one which is needed to complete the covering is $d/2$.

For the randomness gap, consider two instances. In the first one, we have an item of size $\vec{S}(0) = \vec{1} - \vec{e}_k$, k uniformly random, value $v_0 = 1$, and one item of size $\vec{S}(1) = \vec{1}$ and value $v_1 = d$. In the second instance, we have one item of size $\vec{T}(0) = Be(1 - 1/d)\vec{1}$, value $v_0 = 1$, and one item of size $\vec{T}(1) = \vec{1}$ and value $v_1 = d$. Both instances share the same mean item sizes. In the first instance, item 1 is always needed, so the cost of any feasible solution is at least d. In the second instance, however, the optimal policy inserts item 0, and then item 1 with probability $1/d$, for an expected cost of 2. □

Thus using mean item sizes, we cannot prove an approximation factor better than d. We show that this approximation factor can be achieved easily, using a greedy *adaptive* policy.

First, let's consider the problem in dimension 1, where the size of each item is just a Bernoulli random variable. Thus the instance is completely characterized by the mean size values. In this case, a greedy algorithm yields the *optimal solution.*

Lemma 28. *For Stochastic Set Cover of one element, assume the items are ordered, so that*

$$\frac{v_1}{\mu(1)} \leq \frac{v_2}{\mu(2)} \leq \frac{v_3}{\mu(3)} \leq \ldots \frac{v_n}{\mu(n)}$$

(call such an ordering "greedy"). Then inserting items in a greedy ordering yields a covering of minimum expected cost. The adaptivity gap in this case is equal to 1*.*

Proof. First, note that in this setting, adaptivity cannot bring any advantage. Until a feasible solution is obtained, we know that all items must have had size 0. An adaptive policy has no additional information and there is only one possible configuration for every subset of available items. Thus there is an optimal item to choose for each subset of available items and an optimal adaptive policy is in fact a fixed ordering of items.

Consider an ordering $(1, 2, 3, \ldots)$, not necessarily greedy; the expected cost of a feasible solution found by inserting in this order is

$$C = \sum_{k=1}^{n} v_k \prod_{j=1}^{k-1} (1 - \mu(j)).$$

Let's analyze how switching two adjacent items affects C. Note that switching i and $i+1$ affects only the contributions of these two items - the terms corresponding to $k < i$ and $k > i+1$ remain unchanged. The difference in expected cost will be

$$\begin{aligned}
\Delta C &= v_i \left(\prod_{j=1}^{i-1} (1 - \mu(j)) \right) (1 - \mu(i+1)) \; + \; v_{i+1} \left(\prod_{j=1}^{i-1} (1 - \mu(j)) \right) \\
&- \; v_i \left(\prod_{j=1}^{i-1} (1 - \mu(j)) \right) \; - \; v_{i+1} \left(\prod_{j=1}^{i} (1 - \mu(j)) \right) \\
&= \left(\mu(i)\mu(i+1) \prod_{j=1}^{i-1} (1 - \mu(j)) \right) \left(\frac{v_{i+1}}{\mu(i+1)} - \frac{v_i}{\mu(i)} \right).
\end{aligned}$$

Therefore, we can switch any pair of elements such that $\frac{v_i}{\mu(i)} \geq \frac{v_{i+1}}{\mu(i+1)}$ and obtain an ordering whose expected cost has not increased.

Assume that $\mathcal{O}$ is an arbitrary greedy ordering and $\mathcal{O}^*$ is a (possibly different) optimal ordering. If $\mathcal{O} \neq \mathcal{O}^*$, there must be a pair of adjacent items in $\mathcal{O}^*$ which are swapped in $\mathcal{O}$. By switching these two items, we obtain another optimal ordering $\mathcal{O}'$. We repeat this procedure, until we obtain $\mathcal{O}$ which must be also optimal. $\square$

The adaptive greedy algorithm. For Stochastic Set Cover in dimension d, we generalize the greedy algorithm in the following way: For each component $i \in [d]$, we find an optimal ordering restricted only to component i; we denote this by $\mathcal{O}_i$. Then our greedy adaptive algorithm chooses at any point a component j which has not been covered yet, and inserts the next available item from $\mathcal{O}_j$.

Corollary 3. *For Stochastic Set Cover in dimension d, the greedy adaptive policy achieves expected cost*

$$GREEDY \leq d \cdot ADAPT.$$

Proof. When the policy chooses an item from $\mathcal{O}_j$, we charge its cost to a random variable X_j. Note that items from $\mathcal{O}_j$ can be also charged to other variables but an item which is charged to X_j can be inserted only after all items preceding it in $\mathcal{O}_j$ have been inserted already. Thus the value of X_j is at most the cost of covering component j, using the corresponding greedy ordering, and $\mathbf{E}[X_j] \leq ADAPT$. Consequently, $GREEDY = \sum_{i=1}^{d} \mathbf{E}[X_i] \leq d \cdot ADAPT$. □

This algorithm is optimal with respect to the randomness gap, i.e. we cannot improve the lower and upper bounds using only mean item sizes. Possibly, there is a better approximation algorithm but its analysis would have to involve more information on the probability distributions.

Also, this approximation algorithm is adaptive so it doesn't settle the adaptivity gap for Stochastic Set Packing. The final answer is unknown. The best upper bound we can prove is the following.

Theorem 28. *For Stochastic Set Cover,*

$$NONADAPT \leq d^2 \cdot ADAPT$$

and the corresponding non-adaptive policy can be found in polynomial time.

Proof. Consider the greedy ordering $\mathcal{O}_j$ for each component j. We interleave $\mathcal{O}_1, \mathcal{O}_2, \ldots, \mathcal{O}_d$ in the following way: We construct a single sequence $\mathcal{O}^* = (i(1), i(2), i(3), \ldots)$ where $i(t)$ is chosen as the next available item from $\mathcal{O}_{j(t)}$; $j(t)$ to be defined. We set $X_i(0) = 0$ for each $1 \leq i \leq d$, $X_{j(t)}(t) = X_{j(t)}(t-1) + v_{i(t)}$ and $X_k(t) = X_k(t-1)$ for $k \neq j(t)$. In other words, we charge the cost of $i(t)$ to $X_{j(t)}$ which denotes the "cumulative cost" of component $j(t)$. At each time t, we choose the index $j(t)$ in order to minimize $X_{j(t)}(t)$ among all possible choices of $j(t)$.

Consider a fixed component k and the time τ when component k is covered. This is not necessarily by an item chosen from $\mathcal{O}_k$, i.e. $j(\tau)$ doesn't have to be k. If $j(\tau) = k$, denote by q the item from $\mathcal{O}_k$ covering component k: $q = i(\tau)$. If $j(\tau) \neq k$, denote by q the next item to be chosen from $\mathcal{O}_k$ if component k had not been covered yet. We denote by $\mathcal{Q}_k$ the prefix of sequence $\mathcal{O}_k$ up to (and including) item q. We claim that for any j, $X_j(\tau)$ is at most the cost of $\mathcal{Q}_k$: If there is some $X_j(\tau) > v(\mathcal{Q}_k)$, the last item that we charged to X_j should not have been chosen; we should have chosen an item from $\mathcal{O}_k$ which would have kept X_k still bounded by $v(\mathcal{Q}_k)$ and thus

smaller than $X_j(\tau)$. Therefore $X_j(\tau) \leq v(\mathcal{Q}_k)$. For the total cost Z_k spent up to time τ when component k is covered, we get

$$Z_k = \sum_{j=1}^{d} X_j(\tau) \leq d\ v(\mathcal{Q}_k).$$

Now consider the set of items $\mathcal{Q}_k$ which is a prefix of $\mathcal{O}_k$. The probability that $\mathcal{Q}_k$ has length at least l is at most the probability that an (optimal) policy covering component k using the ordering $\mathcal{O}_k$ needs to insert at least l items from $\mathcal{O}_k$; this is the probability that the first $l-1$ items in $\mathcal{O}_k$ attain size 0 in component k. If $ADAPT_k$ denotes the minimum expected cost of an adaptive policy covering component k, we get $\mathbf{E}[v(\mathcal{Q}_k)] \leq ADAPT_k \leq ADAPT$ and $\mathbf{E}[Z_k] \leq d \cdot ADAPT$.

Finally, the total cost spent by our policy is $Z = \max_k Z_k$, since we have to wait for the last component to be covered. Therefore,

$$\mathbf{E}[Z] = \mathbf{E}[\max_{1 \leq k \leq d} Z_k] \leq \sum_{k=1}^{d} \mathbf{E}[Z_k] \leq d^2\ ADAPT.$$

□

8.4 Stochastic Covering with multiplicity

We get better results for Stochastic Covering where each item can be repeated arbitrarily many times. In the following, we do not impose any restrictions on the item sizes. Therefore, we can assume by scaling that we have a unit capacity $\vec{b} = \vec{1}$.

First, we prove that the adaptivity gap for $d = 1$ is between 7/6 and 2.

Lemma 29. *The adaptivity gap for Stochastic Covering with multiplicity in dimension $d = 1$ can be arbitrarily close to 7/6.*

Proof. Consider two types of items, for some small $p, \epsilon > 0$:

- Item 1 has size $S(1) = 0$ with probability $1 - 2p$, $S(1) = \epsilon$ with probability p and $S(1) = 1$ with probability p; value $v_1 = 2p$.
- Item 2 has size $S(2) = 1 - \epsilon$ deterministically and value $v_2 = 1$.

The optimal adaptive policy is: Insert items of type 1 repeatedly until one of them gets nonzero size. If the size is ϵ, complete the solution by adding item 2. The expected cost of this solution is 3/2: 1 for a sequence of expected length $1/2p$ of items of type 1, and then an item of type 2 with probability 1/2.

On the other hand, any non-adaptive policy has expected cost at least 7/4. We assume that ϵ is so small that the probability we accumulate size 1 from $1/\epsilon$ copies of item 1 is negligible, compared to the probability that we get size 1 directly. Inserting an infinite sequence of items of type 1 incurs expected cost $2p \cdot 1/p = 2$ since it takes $1/p$ trials on the average to get an item of size 1.

Inserting two copies of type 2 would cost 2 as well. However, these are not optimal non-adaptive policies.

Consider a non-adaptive policy which inserts a sequence of k copies of type 1, then an item of type 2, and then an infinite sequence of type 1. The probability that we need the j-th item is $(1-p)^{j-1}$ for $j \le k$, and $(1-2p)^{j-2}$ for $j > k$. Thus the expected cost will be

$$\begin{aligned}\mathbf{E}[\text{cost}] &= \sum_{j=1}^{k-1} 2p(1-p)^{j-1} + (1-p)^{k-1} + \sum_{j=k+1}^{\infty} 2p(1-2p)^{j-2} \\ &= 2 - (1-p)^{k-1} + (1-2p)^{k-1}.\end{aligned}$$

For very small p, the optimum choice is $k = \ln 2/p$ and the expected cost tends to $7/4$.

Another possibility is to replace an infinite tail of items of type 1 by an item of type 2. However, in case an item of type 2 has been previously inserted and the policy is still running, this means that the remaining capacity is exactly ϵ. In this case, inserting an item of type 2 has the same effect as inserting an infinite sequence of type 1.

Therefore, the optimal non-adaptive policy has expected cost close to $7/4$ and the adaptivity gap tends to $7/6$. □

Theorem 29. *For Stochastic Covering with multiplicity in dimension $d = 1$, there is a trivial non-adaptive policy (inserting one item type repeatedly) with expected value $NONADAPT \le 2\ ADAPT$.*

Proof. Consider the function $\Psi^+(\vec{b})$ defined in Section 8.1, for $d = 1$. Then it becomes

$$\Psi^+(b) = \min\left\{\sum_i x_i v_i : \ \sum_i x_i \mu(i) \ge b,\ x_i \ge 0\right\}.$$

By Theorem 19, $ADAPT \ge \Psi^+(1)$. Also, in this case $\Psi^+(b)$ has a trivial form: $\Psi^+(b) = \lambda b$, where $\lambda = \min_i(v_i/\mu(i))$. Thus $ADAPT \ge \lambda$.

Now we define our non-adaptive policy: Choose the item type i^* achieving $\lambda = v_{i^*}/\mu(i^*)$ and keep inserting it until capacity 1 is covered. It remains to estimate the expected number of copies needed to do that.

Observe that this policy can also be seen as a Stochastic Knapsack policy which terminates when the capacity of 1 has been exceeded. The only difference is that we count the value of the first overflowing item as well; but Lemma 15 bounds the mass of the set of inserted items $\mathcal{A}$ *including* the overflowing item:

$$\mathbf{E}[\mu(\mathcal{A})] \le 2.$$

Therefore the expected number of items that we insert is at most $2/\mu(i^*)$ and the expected cost of our non-adaptive policy is at most 2λ. □

In general dimension d, a straightforward approach leads to an $O(\log d)$ approximation.

Theorem 30. *For Stochastic Covering with multiplicity in dimension $d \geq 2$,*

$$ADAPT \leq 12 \ln d \; NONADAPT$$

and the corresponding non-adaptive policy can be found in polynomial time.

Proof. Consider the LP formulation of Stochastic Covering with multiplicity:

$$\Psi^+(\vec{b}) = \min \left\{ \sum_i x_i v_i : \; \sum_i x_i \vec{\mu}(i) \geq \vec{b}, \; x_i \geq 0 \right\}.$$

We know from Theorem 19 that $ADAPT \geq \Psi^+(\vec{1})$. We assume by scaling that $\vec{b} = (1, 1, \ldots, 1)$. Let x_i be an optimal solution. We inflate the solution by a factor of $c \ln d$ (hence the need to be able to repeat items) and we build a random multiset $\mathcal{F}$ where item i has an expected number of copies $y_i = x_i \; c \ln d$. This can be done for example by including $\lfloor y_i \rfloor$ copies of item i deterministically and another copy with probability $y_i - \lfloor y_i \rfloor$. Then the total size of set $\mathcal{F}$ in component j can be seen as a sum of independent random $[0, 1]$ variables and the expected total size is

$$\mathbf{E}[S_j(\mathcal{F})] = \sum_i y_i \mathbf{E}[S_j(i)] \geq \sum_i y_i \mu_j(i) \geq c \ln d.$$

By Chernoff bound, with $\mu \geq c \ln d$ and $\delta = 1 - 1/\mu$:

$$\Pr[S_j(\mathcal{F}) < 1] = \Pr[S_j(\mathcal{F}) < (1-\delta)\mu] < e^{-\mu\delta^2/2} \leq e^{-\mu/2+1} \leq \frac{e}{d^{c/2}}.$$

We choose $c = 9$ and then by the union bound

$$\Pr[\exists j; S_j(\mathcal{F}) < 1] < \frac{e}{d^{3.5}}.$$

For $d \geq 2$, $\mathcal{F}$ is a feasible solution with a constant nonzero probability at least $1 - e/2^{3.5}$. Its expected cost is

$$\mathbf{E}[v(\mathcal{F})] = \sum_i y_i v_i = 9 \ln d \; \Psi^+(\vec{1}) \leq 9 \ln d \; ADAPT.$$

If $\mathcal{F}$ is not a feasible solution, we repeat; the expected number of iterations is $1/(1 - e/2^{3.5}) \doteq 1.32$. Therefore

$$NONADAPT \leq 12 \ln d \; ADAPT.$$

This randomized rounding algorithm can be derandomized using pessimistic estimators in the usual way. □

Finally, we show that our $O(\log d)$-approximation for Stochastic Covering with multiplicity is optimal in several ways. We already mentioned the inapproximability result for Set Cover [17] which says that $O(\log d)$ is the best approximation factor we can achieve in polynomial time. In addition to that, it is known that the integrality gap for Set Cover can be $\Omega(\log d)$ [32]. We prove that this implies the same randomness gap for a pair of Stochastic Set Cover instances with multiplicity.

Theorem 31. *If a CIP instance, with item multiplicity, each item size in a domain $\mathcal{D}, \{0,1\}^d \subseteq \mathcal{D} \subseteq [0,1]^d$, unit item values, and capacity $\vec{1} = (1,1,\ldots,1)$, has integrality gap ω, then there is a pair of Stochastic Covering instances with multiplicity, one item type with random size in $\mathcal{D}$, and capacity $\vec{1}$, such that the randomness gap for the two instances is $\rho_a = \rho_n \geq \omega$.*

Proof. Consider such a CIP instance and its corresponding linear relaxation

$$LP = \min\left\{\sum_i x_i : \sum_i x_i \vec{a}(i) \geq \vec{1},\ x_i \geq 0\right\} \tag{8.1}$$

with integrality gap ω. In other words, at least $\omega \cdot LP$ items are required for an integral feasible solution.

Let x_i be an optimal solution, i.e. $\sum x_i = LP$ and $\sum x_i \vec{a}(i) \geq \vec{1}$. We define two Stochastic Covering instances, each with only one item type. In the first one, with probability $p_i = x_i/LP$, let an item have size $\vec{S} = \vec{a}(i)$. The expected size is

$$\vec{\mu} = \mathbf{E}[\vec{S}] = \sum_i \frac{x_i}{LP}\vec{a}(i) \geq \frac{1}{LP}\vec{1}. \tag{8.2}$$

However, whatever the instantiated sizes $\vec{a}(i)$ in a sequence of items, we know that at least $\omega \cdot LP$ items are needed to achieve a covering.

In the second instance, we define an item type with random size $\vec{T}$ with values in $\{0,1\}^d \subseteq \mathcal{D}$: we generate a uniformly random number $\xi \in [0,1]$; for each j, if $\xi \leq \mu_j$, we set $T_j = 1$, otherwise $T_j = 0$. Thus $\mathbf{E}[\vec{T}] = \vec{\mu} = \mathbf{E}[\vec{S}]$. Also, if $T_j = 1$ for the component minimizing μ_j then all the remaining components are equal to 1 as well. We have $\min \mu_j \geq 1/LP$; i.e., each item has size $(1,1,\ldots,1)$ with probability at least $1/LP$. We can insert a sequence of these items until one of them attains size $(1,1,\ldots,1)$ which will take an expected number of at most LP trials.

The same result applies to the non-adaptive randomness gaps since in our instances, there is only one item type and therefore no distinction between adaptive and non-adaptive policies. □

The integrality gap for Set Cover can be achieved with unit item values, for example: Consider a complete k-uniform hypergraph H on $2k$ vertices. I.e. $H = \binom{V}{k}, |V| = 2k$. The *hypergraph vertex cover* problem is to find a set of vertices $W \subseteq V$ intersecting every edge in H. (This is just a reformulation of a Set Cover problem, in a "dual setting".) Obviously, we need $k+1$ vertices to intersect every edge in H. On the other hand, fractionally we can take every vertex with a coefficient of $1/k$ and

this contributes exactly 1 to every edge. The cost of the fractional solution is 2 and the integrality gap for this instance is $(k+1)/2 = \Omega(\log d)$.

By Theorem 31, this means that the randomness gap for Stochastic Set Cover can also be $\Omega(\log d)$. Finally, we show that the adaptivity gap can be $\Omega(\log d)$ as well.

Lemma 30. *There are instances of Stochastic Set Cover with multiplicity where the adaptivity gap is at least* $0.45 \ln d$.

Proof. Consider item types for $i = 1, 2, \ldots, d$ where $\vec{S}(i) = Be(1/2)\ \vec{e}_i$ and $v_i = 1$. An adaptive policy inserts an expected number of 2 items of each type until the respective component is covered; $ADAPT \leq 2d$.

Assume that a nonadaptive policy at some point has inserted k_i items of type i, for each i. Denote the total size at that point by $\vec{S}$. We estimate the probability that this is a feasible solution:

$$\Pr[\vec{S} \geq \vec{1}] = \prod_{i=1}^{d} \Pr[S_i \geq 1] = \prod_{i=1}^{d} (1 - 2^{-k_i}) \leq \exp\left(-\sum_{i=1}^{d} 2^{-k_i}\right).$$

Assume that $k = \sum_{i=1}^{d} k_i = d \log_2 d$. By convexity, the probability of covering is maximized for a given d when $k_i = k/d = \log_2 d$, and then still $\Pr[\vec{S} \geq \vec{1}] \leq 1/e$. Thus whatever the non-adaptive policy does, there is probability $1 - 1/e$ that it needs more than $d \log_2 d$ items, which means $NONADAPT \geq (1 - 1/e) d \log_2 d > 0.9 d \ln d$. $\square$

Chapter 9

Stochastic Optimization and PSPACE

In this chapter, we state several complexity-theoretic results concerning stochastic optimization problems. Specifically, we are interested in Stochastic Knapsack, and its multidimensional generalization Stochastic Packing. Since the deterministic variants of these problems are NP-complete, the stochastic problems must be at least NP-hard. However, it is interesting to ask whether they entail a deeper level of complexity than that of NP. It turns out that many natural questions concerning these problems are actually PSPACE-hard.

9.1 Stochastic Knapsack

Let's begin with Stochastic Knapsack, where items have random sizes and deterministic values, and we consider *adaptive policies* which make decisions based on the instantiated sizes of the items already placed in the knapsack. We exploit the similarity between adaptive policies and *Arthur-Merlin games* [2], or *games against nature* [36], which are both equivalent to PSPACE. Executing an adaptive policy can be seen as a game where Merlin chooses the item to insert, while Arthur responds by selecting a random size for that item. Merlin is trying to prove that a certain kind of solution exists. Once this solution is found, Merlin wins. (And we say that the policy *succeeds* in this case.) If the knapsack overflows, Arthur wins. The probability of Merlin's victory is exactly the probability of success of the optimal adaptive policy. This shows that Stochastic Knapsack can be cast as an Arthur-Merlin game, and therefore it is in PSPACE. However, we prove that various versions of the Stochastic Knapsack and Packing problems are in fact powerful enough to simulate *any* Arthur-Merlin game, thereby proving that answering certain questions about optimal adaptive policies is equivalent to solving anything in PSPACE.

To prove our results, we show reductions from the following PSPACE-complete problem, as described in [12], Fact 4.1:

Problem: *MAX-PROB SSAT*

Input: Boolean 3-cnf formula Φ with variables $x_1, y_1, \ldots, x_k, y_k$.

Regard $\Phi(x_1, y_1, \ldots x_k, y_k)$ as a function from $\{0,1\}^{2k}$ to $\{0,1\}$ and define:

$$P(\Phi) = \mathcal{M}x_1\mathcal{A}y_1\mathcal{M}x_2\mathcal{A}y_2 \ldots \mathcal{M}x_k\mathcal{A}y_k\Phi(x_1, y_1, \ldots x_k, y_k)$$

where $\mathcal{M}x\ f(x) = \max\{f(0), f(1)\}$ and $\mathcal{A}y\ g(y) = (g(0) + g(1))/2$.

Output:

- YES, if $P(\Phi) = 1$.
- NO, if $P(\Phi) \leq 1/2$.

This is a "promise problem", in the sense that any instance is guaranteed to have either $P(\Phi) = 1$ or $P(\Phi) \leq 1/2$. The formula can be seen as encoding a certain Arthur-Merlin game. Specifically, $P(\Phi)$ is the probability that Merlin can satisfy the formula using his optimal strategy in choosing the x_i's, while the y_i's are chosen randomly by Arthur.

We define a Stochastic Knapsack instance which models this Arthur-Merlin game. The reduction is inspired by the standard reduction from 3-SAT to Knapsack.

Theorem 32. *For a Stochastic Knapsack instance and a fixed ordering of the n items $\mathcal{O}$, let $\hat{p}_O$ denote the maximum over all adaptive policies, allowed to insert items only in the order $\mathcal{O}$ (possibly skipping some of them), of the probability that the knapsack is filled exactly up to its capacity. Then it is PSPACE-hard to distinguish whether $\hat{p}_O = 1$ or $\hat{p}_O \leq 1/2^{n^{1-\epsilon}}$, for any fixed $\epsilon > 0$.*

Proof. We show a polynomial-time reduction from *MAX-PROB SSAT* to *Stochastic Knapsack* such that $\hat{p}_O = P(\Phi)$. Suppose that Φ has m clauses, each with at most 3 literals. We define a sequence of items corresponding to the pairs of variables (x_1, y_1), (x_2, y_2), etc. Selecting a particular item will correspond to a choice of value for x_i, and generating a random size will correspond to a choice of value for y_i. Item sizes will be random variables, whose values we write as numbers in representation of base b. We choose b as a constant large enough so that the digits can never overflow ($b = 100$ suffices for all the reductions in this section). The size representation consists of two "blocks":

$$[VARS \mid CLAUSES]$$

where $VARS$ has k digits and $CLAUSES$ has m digits. For each $i \in \{1, 2, \ldots k\}$, we define two items, $I_i(0)$ and $I_i(1)$. Each has two possible sizes, occurring with probability $1/2$, $size(I_i(x_i), 0)$ or $size(I_i(x_i), 1)$:

$$size(I_i(x_i), y_i) = [VARS(i) \mid CLAUSES(i, x_i, y_i)]$$

where $VARS(i)$ has a 1 in the i-th digit and zeros otherwise, and $CLAUSES(i, a, b)$ has 1's marking the clauses satisfied by setting $x_i = a$, $y_i = b$. We also define two "fill-in" items $F_j(0), F_j(1)$ for each clause j, with

$$size(F_j(x)) = [0 \mid CLAUSE(j)],$$

where $CLAUSE(j)$ has a 1 in the j-th digit and zeros otherwise. The fixed ordering $\mathcal{O}$ is

$$(I_1(0), I_1(1), I_2(0), I_2(1), \ldots, F_1(0), F_1(1), \ldots)$$

and we set the knapsack capacity to

$$C = [1111111111 \mid 3333333333333333333].$$

For any instance, we can define an adaptive policy that succeeds with probability $P(\Phi)$. Let $P(\Phi) = \mathcal{M}x_1\mathcal{A}y_1 f_1(x_1, y_1)$; there is a value x_1^* such that $P(\Phi) = (f_1(x_1^*, 0) + f_1(x_1^*, 1))/2$. Our adaptive policy chooses $I_1(x_1^*)$ as its first item. Its size is chosen randomly from one of two possibilities, which correspond to $y_1 = 0$ or $y_1 = 1$. The ensuing policy will depend on which of these two alternatives occurs, and we want the final probability of success to be the average of $f_1(x_1^*, 0)$ and $f_1(x_1^*, 1)$. Therefore, we would like to ensure that the probability of success, conditioned on a fixed value of y_1, will be $f_1(x_1^*, y_1)$. So fix this value to be $y_1 = y_1^*$. Now we have $size(I_1(x_1^*), y_1^*)$ in the knapsack, which corresponds to marking the clauses satisfied by x_1^* and y_1^*. We want to continue the policy so that we succeed with probability $f_1(x_1^*, y_1^*)$. By further expansion of $f_1(x_1^*, y_1^*) = \mathcal{M}x_2\mathcal{A}y_2 f_2(x_2, y_2)$, there exists x_2^* such that $(f_2(x_2^*, 0) + f_2(x_2^*, 1))/2 = f_1(x_1^*, y_1^*)$. Item $I_2(x_2^*)$ will be the next one to be chosen, etc. It can be seen that if the sequence of variables is chosen in such a way that $\Phi(x_1^*, y_1^*, \ldots x_k^*, y_k^*) = 1$, then the selection of items produces a digit 1 in each position of $VARS$ and a digit between 1 and 3 in each position of $CLAUSES$. Then we can insert some of F_j if necessary, to ensure that every digit of $CLAUSES$ is equal to 3 and the policy succeeds. By unfolding the formula, it is clear that the total probability of success is $P(\Phi)$. Therefore $\hat{p}_O \geq P(\Phi)$.

Conversely, there can be no adaptive policy which succeeds with probability more than $P(\Phi)$. Consider an optimal adaptive policy, which inserts items $I_1(x_1), I_2(x_2), \ldots$ in this order, one of each index (otherwise the total size in the $VARS$ block cannot attain precisely 1111111111). The choices of items $I_i(x_i)$ in response to the perceived sizes correspond to a strategy of setting the x_i variables in response to the values of y_i. The adaptive policy can succeed only if each digit of $CLAUSES$ is at least 1 after inserting $I_1(x_1), I_2(x_2), \ldots, I_k(x_k)$; otherwise it cannot achieve the full capacity even with the fill-in items. This means that with probability $\hat{p}_O$, all clauses are satisfied by the respective assignment of $(x_i, y_i)_{i=1}^n$, i.e. $P(\Phi) \geq \hat{p}_O$.

Thus, in case of a YES instance, $\hat{p}_O = 1$, and in case of a NO instance $\hat{p}_O \leq 1/2$. Finally, we show how to boost this gap to an exponentially large one. We choose a polynomial n^q and we concatenate n^q copies of our instance, using parallel blocks of digits with sufficient margins of zero digits, to prevent overflow between blocks. The items corresponding to each copy will affect only the respective block of digits.

The target capacity will be $C^* = [C|C|\ldots|C]$. An adaptive policy can achieve this capacity exactly if and only if it achieves capacity C in each copy independently. This happens with probability $(\hat{p}_O)^{n^q}$ which is still 1 in case of a YES instance, and at most $1/2^{n^q}$ in case of a NO instance. The number of new items is $n' = n^{q+1}$, which means that $n^q = (n')^{1-1/(q+1)}$ and this holds for an arbitrarily large constant q. It follows that the probability of success is either 1 or at most $\leq 1/2^{n^{1-\epsilon}}$. □

Theorem 33. *For a knapsack instance with n items, let $\hat{p}$ be the maximum probability that an adaptive policy fills the knapsack exactly to its capacity. Then it is PSPACE-hard to distinguish whether $\hat{p} = 1$ or $\hat{p} \leq 1/2^{n^{1-\epsilon}}$.*

Proof. We already know that the problem is PSPACE-hard if a fixed ordering is imposed on the items. We show how to remove this restriction here. Consider the knapsack instance generated in the previous proof, where the optimal policy has probability of success either $\hat{p}_O = 1$ or $\hat{p}_O \leq 1/2$. We extend the item sizes by a new block called $FLAGS$ with k digits which will enforce that the optimal policy must consider the items $I_1, I_2, I_3, \ldots$ in this order. The fill-in items F_j receive a zero $FLAGS$ block - the adaptive policy cannot gain anything by inserting them earlier anyway. $FLAGS$ will have a digit for each index i and each item $I_i(x)$ will be modified in this way: we define two new items for each item, parametrized by $f_i = 0, 1$, and two new random sizes for each previously possible size, parametrized by $r_i = 0, 1$. The new item sizes will be $size(I_i(x_i, f_i), y_i, r_i)) =$

$$[VARS(i) \mid FLAGS(i, f_i, r_i) \mid CLAUSES(i, x_i, y_i)]$$

where $VARS$ and $CLAUSES$ are defined as before, and

$$FLAGS(i, f_i, r_i) = [0000000 r_i f_i 000000]$$

where f_i is the i-th least significant digit and r_i is the $(i+1)$-th least significant digit. (The items for $i = k$ don't have any r_i digit.) Every $I_i(x_i, f_i)$ item can therefore take 4 different values. Finally, we extend the knapsack capacity to

$$C = [1111111111 \mid 1111111111 \mid 333333333333333].$$

Consider a YES instance, in which case $\hat{p}_O = 1$. Then an optmal policy which inserts items in the standard order $I_1, I_2, I_3, \ldots$, can always choose a suitable value of f_{i+1} to ensure $r_i + f_{i+1} = 1$, which achieves the target value for $FLAGS$. Therefore an optimal policy succeeds with probability $\hat{p} = \hat{p}_O = 1$.

Now consider a NO instance, where $\hat{p}_O \leq 1/2$. A policy may ignore the standard ordering and try to insert an item $I_i(x_i, f_i)$ *after* $I_{i+1}(x_{i+1}, f_{i+1})$. Let's say that the policy "cheats" in this case (which may happen in certain branches of the decision tree). Since the digit r_i is generated randomly, and the only other non-zero entry in this position is f_{i+1}, which has been chosen already, the final entry in this position will be 1 only with probability 1/2. Therefore the branches where a policy cheats can succeed with conditional probability at most 1/2. Suppose that the policy cheats with total probability p_C, and the optimal ordered policy succeeds with probability

$\hat{p}_O \leq 1/2$. Then the cheating policy can achieve maximum probability of success when $p_C = 1 - \hat{p}_O$ and that is $\hat{p} \leq \hat{p}_O + (1 - \hat{p}_O)/2 \leq 3/4$. Using the method from the previous proof, we can then reduce the probability of success to $1/2^{n^{1-\epsilon}}$ for any fixed $\epsilon > 0$. □

Now we would like to extend the hardness results to the optimization variant of the problem, where we look for an adaptive policy maximizing the expected value obtained before the knapsack overflows. This is possible to accomplish quite easily, if we allow random item values correlated with the random sizes (see below). Unfortunately, the following note indicates that PSPACE-hardness cannot be obtained with deterministic values.

Note: The following problem is in NP: *Given a Stochastic Knapsack instance with random (discrete) sizes and deterministic values, decide whether it is possible to achieve value at least V with probability 1 or not.*

Indeed, we can replace each random size by its maximum that can occur with nonzero probability, and solve the corresponding knapsack problem. It is possible to achieve value V with probability 1 if and only it is possible in the worst possible scenario where every item gets its maximum size.

Now we turn to the more general setting with random values.

Theorem 34. *For a Stochastic Knapsack instance, where item values are random and possibly correlated with the respective item sizes, let $\hat{p}(V)$ be the maximum probability that an adaptive policy inserts successfully a set of items of total value at least V. Similarly, define $\hat{p}_O(V)$ for ordered policies with a given ordering of items $\mathcal{O}$. Then it is PSPACE-hard to distinguish whether $\hat{p}(V)$ (or $\hat{p}_O(V)$, resp.) is equal to 1 or at most $1/2^{n^{1-\epsilon}}$.*

Proof. In the setting with random values, we can define each to be equal to the size of the respective item. Then the maximum value any policy can achieve is equal to the capacity, and the maximum probability that this value is attained is $\hat{p}(C) = \hat{p}$ (from Theorem 33), or $\hat{p}_O(C) = \hat{p}_O$ (from Theorem 32). Consequently, deciding between $\hat{p} = 1$ and $\hat{p} \leq 1/2^{n^{1-\epsilon}}$ is PSPACE-hard, as well as deciding between $\hat{p}_O = 1$ and $\hat{p}_O \leq 1/2^{n^{1-\epsilon}}$ for any fixed ordering of items. □

Theorem 35. *For a Stochastic Knapsack instance with random values, possibly correlated with item sizes, it is PSPACE-hard to maximize the expected value achieved by any adaptive policy, or by any adaptive policy with a fixed ordering of items.*

Proof. We use the reduction from the proof of Theorem 34. In case of a YES instance, the expected value achieved by the optimal adaptive policy is C, because C is achieved with probability 1, and it is the maximum possible value for any policy. In case of a NO instance, the probability of achieving C is less than $1/2$; with probability at least $1/2$, the policy achieves at most value $C - 1$. Thus the expected value in this case cannot be larger than $C - 1/2$. The same holds, if we restrict ourselves to policies using the standard ordering of items. If we could optimize the expected value achievable in either case, we could solve *MAX-PROB SSAT* in polynomial time. □

9.2 Stochastic Packing

In the following, we turn to the complexity of multidimensional Stochastic Knapsack, i.e. general stochastic packing. In this setting, sizes are random vectors in $\Re^d$, and values are deterministic scalars. In contrast to the inapproximability results of Chapter 7, here we do not regard d as part of the input. Let us note that for deterministic packing problems, it is NP-hard to find the optimum but there is a PTAS for any fixed $d \geq 1$ [19].

Theorem 36. *For Stochastic Packing in fixed dimension $d \geq 2$, let $\hat{p}(V)$ be the maximum probability that an adaptive policy inserts successfully a set of items of total value at least V. Then for any fixed $\epsilon > 0$, it is PSPACE-hard to distinguish whether $\hat{p}(V) = 1$ or $\hat{p}(V) \leq 1/2^{n^{1-\epsilon}}$. For any fixed $\xi \in (0, 1]$, it is also PSPACE-hard to maximize V subject to the condition that $\hat{p}(V) \geq \xi$.*

Proof. We can assume that $d = 2$. We define items of the same types as we used in the preceding proofs, for a 3-cnf formula with variables $x_1, y_1, \ldots, x_k, y_k$ and m clauses. The 2-dimensional sizes will have the same format in each component, $[FLAGS \mid CLAUSES]$, where $FLAGS$ have k digits and $CLAUSES$ have m digits. It will be convenient to consider the actual digits as "2-dimensional", with a pair of components. In addition, we define item values with a similar format $[VARS \mid CLAUSES]$ where $VARS$ have k digits and $CLAUSES$ have m digits. The variable items are defined as follows:

$$\vec{s}(I_i(x_i, f_i), y_i, r_i) = [FLAGS(i, f_i, r_i) \mid CLAUSES(i, x_i, y_i)]$$

where $FLAGS(i, f_i, r_i)$ have two nonzero digits: the i-th most significant digit has a 1 in the f_i-component and the $(i + 1)$-th most significant digit has a 1 in the r_i-component, except for $i = 1$ where $I_1(x_1, f_1)$ puts 1 in both components of the first digit, and for $i = k$ where $I_k(x_k, f_k)$ doesn't have any random digit r_k. In $CLAUSES(i, x_i, y_i)$, x_i adds 1 to one of the two components of each digit corresponding to a clause in which x_i appears, and it will be component 1 if the clause is satisfied by the value of x_i, or component 2 if the clause is not satisfied. Similarly, y_i adds 1 to the digits corresponding to its appearance in clauses. If a clause is satisfied or unsatisfied by both variables simultaneously, the respective component will receive a contribution of 2. The values are defined as

$$val(I_i(x_i, f_i)) = [VARS(i) \mid 0]$$

where $VARS(i)$ contains a 1 in the i-th digit and zeros otherwise.

Then, we define fill-in items F_{ji}, which put a 1 in the i-component of the j-th clause. For each j, we have 2 items of type F_{j1} and 3 items of type F_{j2}. Their values are

$$val(F_{ji}) = [0 \mid CLAUSE(j)]$$

i.e., a 1 marking the j-th clause. The capacity of the knapsack is

$$C = [11111111 \mid 33333333333333]$$

in each dimension and the target value is also

$$V = [11111111 \mid 33333333333333].$$

Assume that $P(\Phi) = 1$. This implies an adaptive policy which inserts one item I_i for each i in order, choosing $f_{i+1} = 1 - r_i$ for each $i < k$. Based on the satisfying strategy for formula Φ, the policy satisfies each clause and then adds fill-in items to achieve value 1 in each digit of $VARS$ and value 3 in each digit of $CLAUSES$.

On the other hand, assume $P(\Phi) \leq 1/2$. First, note that assuming that at most 1 item I_i for each i is inserted, a policy must insert *at least* 1 item I_i for each i. Otherwise it cannot achieve the target value in the $VARS$ block which is the most significant block. Next, any adaptive policy inserting exactly 1 item I_i for each i and abiding by the standard ordering of items can achieve the target value only if all clauses are properly satisfied (because otherwise it would need 3 items of type F_{j1} for some clause), and that happens with probability at most 1/2.

However, we have to be careful about "cheating policies" just like before. Here, "cheating" means either inserting I_i after I_{i+1} or not inserting exactly 1 copy of each I_i. Consider a cheating policy and the first i for which this happens. In case I_i is not inserted at all, the policy cannot achieve the target value for $VARS$. In case more than 1 copy of I_i is inserted or I_i is inserted after I_{i+1}, there is 1/2 probability of overflow in the $FLAGS$ block of capacity. Either way, this leads to a failure with probability at least 1/2, conditioned on the event of cheating. Therefore the probability of success of a cheating policy is at most 3/4. Finally, we can reduce this probability to $1/2^{n^{1-\epsilon}}$ by the method of parallel concatenation (see Theorem 32).

We also get PSPACE-hardness for the question of maximizing V subject to $\hat{p}(V) \geq \xi$. This is because for large enough n we could distinguish between $\hat{p}(V) = 1$ and $\hat{p}(V) < 1/2^{n^{1-\epsilon}}$. □

Theorem 37. *For a 2-dimensinal Stochastic Knapsack instance, it is PSPACE-hard to maximize the expected value achieved by an adaptive policy.*

Proof. We use the reduction from the previous proof. The maximum value that any policy can achieve is $V = [11111111 \mid 33333333333333]$. In case of a YES instance, an optimal policy achieves V with probability 1, whereas in case of a NO instance, it can succeed only with probability less than 1/2. Therefore the expected value obtained in this case is at most $V - 1/2$. □

9.3 Stochastic Covering

We get the same results for Stochastic Covering, without multiplicity. For Theorems 32 and 33, it is clear that we are not using the packing condition - the question is only whether an adaptive policy can fill the knapsack *exactly* to its capacity. As a consequence, we get statements analogous to Theorems 34 and 35 for Stochastic Covering.

Theorem 38. *For a Stochastic Covering instance in 1 dimension, where item values are random and possibly correlated with the respective item sizes, let $\hat{p}(V)$ be the maximum probability that an adaptive policy finds a feasible covering of total value at most V. Similarly, define $\hat{p}_O(V)$ for ordered policies with a given ordering of items $\mathcal{O}$. Then it is PSPACE-hard to distinguish whether $\hat{p}(V)$ (or $\hat{p}_O(V)$, resp.) is equal to 1 or at most $1/2^{n^{1-\epsilon}}$.*

Theorem 39. *For a Stochastic Covering instance in 1 dimension with random values, possibly correlated with item sizes, it is PSPACE-hard to minimize the expected value of a covering achievable by an adaptive policy, or by an adaptive policy with a fixed ordering of items.*

For Stochastic Covering in 2 dimensions, we get results similar to Theorems 36 and 37.

Theorem 40. *For Stochastic Covering in fixed dimension $d \geq 2$, let $\hat{p}(V)$ be the maximum probability that an adaptive policy finds a feasible covering of total value at most V. Then for any fixed $\epsilon > 0$, it is PSPACE-hard to distinguish whether $\hat{p}(V) = 1$ or $\hat{p}(V) \leq 1/2^{n^{1-\epsilon}}$. For any fixed $\xi \in (0, 1]$, it is also PSPACE-hard to maximize V subject to the condition that $\hat{p}(V) \geq \xi$.*

Theorem 41. *For a 2-dimensional Stochastic Covering instance, it is PSPACE-hard to maximize the expected value achieved by an adaptive policy.*

Note that our reductions do not work if we allow item multiplicity. We do not know whether these questions are PSPACE-hard with item multiplicity as well.

Bibliography

[1] D. Applegate and W. Cook. Solving large-scale matching problems. In *D. Johnson and C.C. McGeoch, eds., Network Flows and Matchings*, 1993.

[2] L. Babai and S. Moran. Arthur-Merlin games: a randomized proof system, and a hierarchy of complexity classes. *J. Computer and System Sciences*, 36(2):254–276, 1988.

[3] B.Bollobás. *Extremal graph theory*. Academic Press, London-New York, 1978.

[4] D. Bertsimas. The probabilistic minimum spanning tree. *Networks*, 20:245–275, 1990.

[5] J.R. Birge and F.V. Louveaux. *Introduction to stochastic programming*. Springer Verlag, 1997.

[6] B. Bollobás and A. Thomason. Threshold functions. *Combinatorica*, 7:35–38, 1987.

[7] Robert D. Carr, Lisa K. Fleischer, Vitus J. Leung, and Cynthia A. Phillips. Strengthening integrality gaps for capacitated network design and covering problems. In *SODA*, pages 106–115, 2000.

[8] R.L. Carraway, R.L.Schmidt, and L.R. Weatherford. An algorithm for maximizing target achievement in the stochastic knapsack problem with normal returns. *Naval Research Logistics*, 40:161–173, 1993.

[9] B. Chazelle. A minimum spanning tree algorithm with inverse-Ackermann type complexity. *JACM*, 47:1028–1047, 2000.

[10] C. Chekuri and S. Khanna. On multidimensional packing problems. *SIAM J. Computing*, 33(4):837–851, 2004.

[11] V. Chvátal. A greedy heuristic for the set covering problem. *Math. Oper. Res*, 4:233–235, 1979.

[12] A. Condon, J. Feigenbaum, C. Lund, and P. Shor. Random debaters and the hardness of approximating stochastic functions. *SIAM J. Computing*, 26:369–400, 1997.

[13] D. Peleg, A.A. Schäffer. Graph spanners. *J. Graph Theory*, 13(1):99–116, 1989.

[14] B. Dean. *Approximation algorithms for stochastic scheduling problems*. PhD thesis, Massachusetts Institute of Technology, 2005.

[15] B. Dean, M. X. Goemans, and J. Vondrák. Approximating the stochastic knapsack: the benefit of adaptivity. In *FOCS*, pages 208–217, 2004.

[16] C. Derman, C.J. Lieberman, and S.M. Ross. A renewal decision problem. *Management Science*, 24(5):554–561, 1978.

[17] U. Feige. A threshold of ln n for approximating set cover. *JACM*, 45(4):634–652, 1998.

[18] U. Feige. On sums of independent random variables with unbounded variance, and estimating the average degree in a graph. In *STOC*, pages 594–603, 2004.

[19] A.M. Frieze and M.R.B. Clarke. Approximation algorithms for the m-dimensional 0-1 knapsack problem: Worst-case and probabilistic analyses. *European J. Oper. Res.*, 15:100–109, 1984.

[20] A. Goel and P. Indyk. Stochastic load balancing and related problems. In *FOCS*, pages 579–586, 1999.

[21] A. Gupta, M. Pál, R. Ravi, and A. Sinha. Boosted sampling: approximation algorithms for stochastic optimization. In *SODA*, pages 417–426, 2004.

[22] M. Halldórsson. Approximations of weighted independent set and subset problems. *J. of Graph Algorithms and Applications*, 4(1):1–16, 2000.

[23] M. Henig. Risk criteria in a stochastic knapsack problem. *Operations Research*, 38(5):820–825, 1990.

[24] J. Holm, K. de Lichtenberg, and M. Thorup. Poly-logarithmic deterministic fully-dynamic algorithms for connectivity, minimum spanning tree, 2-edge, and bicoonectivity. *J. ACM*, 48(4):723–760, 2001.

[25] N. Immorlica, D. Karger, M. Minkoff, and V. Mirrokni. On the costs and benefits of procrastination: Approximation algorithms for stochastic combinatorial optimization problems. In *SODA*, pages 184–693, 2004.

[26] N. Jing, Y.W. Huang, and E.A. Rundensteiner. Hierarchical encoded path views for path query processing: An optimal model and its performance evaluation. *IEEE T. on Knowledge and Data Engineering*, 10(3):409–432, 1998.

[27] D. Johnson. Approximation algorithms for combinatorial problems. *J. Comp. Syst. Sci.*, 9:256–216, 1974.

[28] D.R. Karger, P.N. Klein, and R.E. Tarjan. A randomized liner-time algorithm to find minimum spanning trees. *JACM*, 42(2):321–328, 1995.

[29] J. Kleinberg, Y. Rabani, and E. Tardos. Allocating bandwidth for bursty connections. In *STOC*, pages 664–673, 1997.

[30] A. Kleywegt and J.D. Papastavrou. The dynamic and stochastic knapsack problem with random sized items. *Operations Research*, 49(1):26–41, 2001.

[31] S. Kolliopoulos and N. Young. Tight approximation results for general covering integer programs. In *FOCS*, page 522, 2001.

[32] L. Lovász. On the ratio of the optimal integral and fractional covers. *Disc. Math.*, 13:256–278, 1975.

[33] L.Valiant. The complexity of enumeration and reliability problems. *SIAM J. on Computing*, 8:410–421, 1979.

[34] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[35] P. Erdös and H. Sachs. Reguläre Graphen gegebener Taillenweite mit minimaler Knotenzahl. *Wiss Z Martin-Luther University Halle-Wittenberg Math-Natur*, Reihe 12:251–257, 1963.

[36] C.H. Papadimitriou. Games against nature. *J. of Computer and System Sciences*, 31:288–301, 1985.

[37] J.D. Papastavrou, S. Rajagopalan, and A. Kleywegt. The dynamic and stochastic knapsack problem with deadlines. *Management Science*, 42(12):1706–1718, 1996.

[38] R. Aharoni and P. Erdös and N. Linial. Optima of dual integer linear programs. *Combinatorica*, 8:13–20, 1988.

[39] P. Raghavan. Probabilistic construction of deterministic algorithms: approximating packing integer programs. *J. Comp. and System Sci.*, 37:130–143, 1988.

[40] P. Raghavan and C. D. Thompson. Randomized rounding: a technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7:365–374, 1987.

[41] R.L. Graham, Martin Grötschel and László Lovász. *Handbook of combinatorics*. Elsevier, 1995.

[42] M. Rothkopf. Scheduling with random service times. *Management Science*, 12(9):707–713, 1966.

[43] D. Shmoys and C. Swamy. Stochastic optimization is (almost) as easy as deterministic optimization. In *FOCS*, pages 228–237, 2004.

[44] M. Sniedovich. Preference order stochastic knapsack problems: methodological issues. *The Journal of the Operational Research Society*, 31(11):1025–1032, 1980.

[45] A. Srinivasan. Improved approximations of packing and covering problems. In *STOC*, pages 268–276, 1995.

[46] J.Håstad. Clique is hard to approximate to within $n^{1-\epsilon}$. In *FOCS*, pages 627–636, 1996.

[47] E. Steinberg and M.S. Parks. A preference order dynamic program for a knapsack problem with stochastic rewards. *The Journal of the Operational Research Society*, 30(2):141–147, 1979.

[48] M. Uetz. *Algorithms for deterministic and stochastic scheduling*. PhD thesis, Institut für Mathematik, Technische Universität Berlin", 2001.