Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών

Προσεγγιστικοί Αλγόριθμοι για Συνδυαστικά Προβλήματα Στοχαστικής Βελτιστοποίησης

Διπλωματική Εργασία

του

Αγγελιδάκη Χαράλαμπου

Επιβλέπων:	Ευστάθιος Ζάχος Καθηγητής Ε.Μ.Π.
Συνεπιβλέπων:	Δημήτρης Φωτάκης Λέκτορας Ε.Μ.Π.

Εργαστήριο Λογικής και Επιστήμης Υπολογισμών
Αθήνα, Μάρτιος 2011

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών

Προσεγγιστικοί Αλγόριθμοι για Συνδυαστικά Προβλήματα Στοχαστικής Βελτιστοποίησης

Διπλωματική Εργασία

του

Αγγελιδάκη Χαράλαμπου

Επιβλέπων:	Ευστάθιος Ζάχος Καθηγητής Ε.Μ.Π.
Συνεπιβλέπων:	Δημήτρης Φωτάκης Λέκτορας Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την $14^{η}$ Μαρτίου 2011.

....	....	....
Ευστάθιος Ζάχος	Άρης Παγουρτζής	Δημήτρης Φωτάκης
Καθηγητής Ε.Μ.Π.	Επίκουρος Καθηγητής Ε.Μ.Π.	Λέκτορας Ε.Μ.Π.

Εργαστήριο Λογικής και Επιστήμης Υπολογισμών
Αθήνα, Μάρτιος 2011

....
Χαράλαμπος Αγγελιδάκης
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Ο σκοπός της διπλωματικής αυτής εργασίας είναι η μελέτη αλγορίθμων για μια ενδιαφέρουσα παραλλαγή προβλημάτων συνδυαστικής βελτιστοποίησης στην οποία δε γνωρίζουμε με ακρίβεια τα δεδομένα εισόδου, αλλά γνωρίζουμε μόνο μία πιθανοτική κατανομή πάνω σε αυτά. Τέτοια προβλήματα εμπίπτουν στο ευρύτερο πεδίο της Στοχαστικής Βελτιστοποίησης. Σκοπός της Στοχαστικής Βελτιστοποίησης είναι η μοντελοποίηση της αβεβαιότητας στα δεδομένα εισόδου ενός προβλήματος. Η λήψη αποφάσεων σε συνθήκες αβεβαιότητας για τις μελλοντικές απαιτήσεις προκύπτει συχνά σε πολλές περιοχές, όπως σε προβλήματα σχεδιασμού δικτύων οποιασδήποτε φύσης, στα οποία δεν είναι ρεαλιστικό να θεωρήσουμε ότι γνωρίζουμε εξαρχής το ακριβές σύνολο των απαιτήσεων μας. Σε τέτοιες περιπτώσεις λοιπόν, οι απαιτήσεις δε μας είναι γνωστές με ακρίβεια, αλλά μέσω μοντέλων προσομοίωσης, ερευνών και μοντέλων προβλέψεων, μπορούμε να αποκτήσουμε στατιστικά δεδομένα για αυτές. Επίσης, αν και μπορούμε να πάρουμε πιο αποτελεσματικές αποφάσεις όταν λάβουμε τις ακριβείς πληροφορίες, το κόστος των αποφάσεων αυτών θα είναι τότε πιο ακριβό. Αυτό αντανακλά την απόλυτα λογική αύξηση του κόστους για αποφάσεις που πρέπει να παρθούν και να υλοποιηθούν πολύ άμεσα, σε αντίθεση με τις αποφάσεις που μπορούμε να λάβουμε εκ των προτέρων και οι οποίες υλοποιούνται σε μεγαλύτερο βάθος χρόνου.

Η πληροφορία σε τέτοια προβλήματα μας δίνεται σταδιακά, σε διακριτές στιγμές στο χρόνο που τις ονομάζουμε στάδια (stages). Θα ασχοληθούμε αρχικά με προβλήματα 2 σταδίων (2-stage), τα οποία αποτελούν την αφετηρία για οποιονδήποτε θέλει να ασχοληθεί με το πεδίο αυτό, και στη συνέχεια με προβλήματα πολλών σταδίων (multistage), και θα ορίσουμε τα προβλήματα βελτιστοποίησης μας σε σχέση με τη μέση τιμή του κόστους που προκύπτει σε όλα τα στάδια. Πιο συγκεκριμένα, ο στόχος μας είναι να κάνουμε την καλύτερη δυνατή επιλογή στο πρώτο στάδιο ώστε η μέση τιμή του κόστους που θα προκύψει σε όλα τα στάδια να είναι η ελάχιστη δυνατή. Ένα πολύ σημαντικό σημείο σε τέτοια προβλήματα είναι το πως μας δίνονται οι πιθανοτικές πληροφορίες. Θα επικεντρωθούμε στο λεγόμενο *black-box* μοντέλο, το οποίο μας δίνει τη δυνατότητα να μοντελοποιήσουμε αυθαίρετες κατανομές με κάθε είδους συσχετισμούς.

Την τελευταία δεκαετία έχει γίνει αρκετή δουλειά στην περιοχή αυτή και θα παρουσιάσουμε ένα μέρος αυτής. Η δειγματοληψία (sampling) θα παίξει κεντρικό ρόλο καθώς είναι συνυφασμένη με το black-box μοντέλο. Θα χρησιμοποιήσουμε πολλές και διαφορετικές τεχνικές από πεδία όπως η θεωρία πιθανότητας και η γραμμική και κυρτή βελτιστοποίηση ώστε να προσεγγίσουμε τα προβλήματα αυτά. Προσπαθήσαμε να κρατήσουμε τις μεθόδους που παρουσιάζουμε όσο πιο γενικές γίνεται, με σύντομες αναφορές στις εφαρμογές τους, κυρίως σε προβλήματα κάλυψης (Set Cover, Vertex Cover) και προβλήματα χωροθέτησης υπηρεσιών (Facility Location Problems). Ελπίζουμε ότι το υλικό που θα παρουσιάσουμε θα επιτρέψει στον αναγνώστη να αποκτήσει μια καλή εικόνα για τα συνδυαστικά προβλήματα στοχαστικής βελτιστοποίησης και να κατανοήσει τις εγγενείς δυσκολίες που παρουσιάζουν και που πηγάζουν από το πιθανοτικό κομμάτι της εισόδου.

Abstract

The purpose of this thesis is the study of algorithms for an interesting class of variants of combinatorial optimization problems in which we do not know the exact input, but only probabilistic information about it. Such problems fall under the broad field of Stochastic Optimization. Stochastic Optimization attempts to model uncertainty in the input data. The concept of having to make decisions under uncertainty about future requirements rises naturally in many areas, including transportation models, logistics, financial instruments and network design, where it is unrealistic to consider that the exact input is known in advance. In such scenarios, the demand pattern is not known precisely at the outset, but one might be able to obtain, through simulation models, surveys or market predictions, statistical information about the demands. Also, while more effective decisions can be made when the actual requirements are given, the decision-making costs are inflated until then. This reflects the increased cost of rapid-response, as it is reasonable to assume that decisions that need to be implemented in short time are much costlier than decisions that are implemented during longer periods of time.

Information in such problems is usually revealed in discrete moments in time, which we call stages. We will focus on both 2-stage problems, which are the starting point for anyone interested in the field, and multistage problems, and define our optimization problems in terms of the expected cost incurred in all stages, i.e. our goal is to make the best possible choice in the first-stage so that the expected cost incurred in all stages is as low as possible. A crucial point in such problems is how the probabilistic information is revealed. We will focus on the so-called *black-box* model, which allows us to model arbitrary distributions with any kinds of correlations.

Much work has been done the last decade in the field and we will present a part of it. Sampling will play central part as it goes hand-in-hand with the black-box model. Techniques from various fields such as probability theory and linear and convex optimization theory will be utilized in order to make these generally hard problems more tractable. We have tried to keep the methods presented as generic as possible, with a brief discussion of their applications, mostly on covering problems (Set Cover, Vertex Cover) and Facility Location Problems. We hope that the presentation material will allow the reader to gain an insight of stochastic combinatorial optimization problems and the inherent difficulties that stem from the probabilistic part of the input.

Keywords

stochastic combinatorial optimization, sampling, black-box model, boosted sampling framework, sample average approximation, Steiner Tree, Facility Location, Set Cover, Vertex Cover

Ευχαριστίες

Με την ολοκλήρωση της διπλωματικής αυτής εργασίας και των προπτυχιακών μου σπουδών στο Εθνικό Μετσόβιο Πολυτεχνείο, θα ήθελα να ευχαριστήσω όλους τους καθηγητές μου και τους ανθρώπους του ιδρύματος που με βοήθησαν όλα αυτά τα χρόνια.

Ιδιαίτερα θα ήθελα να απευθυνθώ και να ευχαριστήσω θερμά τα μέλη του Εργαστηρίου Λογικής και Επιστήμης Υπολογισμών (Corelab). Τον καθηγητή μου κ. Ζάχο, στον οποίο οφείλω πολλά. Ο ενθουσιασμός του και η αμεσότητα του με τους φοιτητές, σε συνδυασμό με το δεδομένο επιστημονικό του κύρος υπήρξαν καθοριστικοί παράγοντες για να αγαπήσω και εγώ τη Θεωρητική Πληροφορική. Τον καθηγητή μου κ. Φωτάκη, ο οποίος με την πάντα καλή του διάθεση, την υπομονή του, την καθοδήγηση καθ' όλη τη διάρκεια της εργασίας αυτής, τις πολύ χρήσιμες συμβουλές και τη διαρκή ενθάρρυνση από μέρους του με βοήθησε σημαντικά τον τελευταίο χρόνο των σπουδών μου και χωρίς τον οποίο η διπλωματική αυτή δε θα μπορούσε να γίνει. Επίσης, τον καθηγητή μου κ. Παγουρτζή, κοντά στον οποίο μαθαίνω από το 1ο εξάμηνο μέχρι και σήμερα, και που έχει παίξει βασικό ρόλο στη διαμόρφωση της επιστημονικής μου σκέψης. Φυσικά, δε θα μπορούσα να ξεχάσω και όλους τους φοιτητές του εργαστηρίου, με τους οποίους έχουμε περάσει τόσες ωραίες στιγμές, και που με το πολύ καλό κλίμα που έχουν δημιουργήσει στο εργαστήριο κάνουν πιο ευχάριστη την εργασία σε αυτό.

Επίσης, θα ήθελα να ευχαριστήσω όλους τους φίλους και ανθρώπους, εντός και εκτός σχολής, που με έχουν στηρίξει όλα αυτά τα χρόνια και με έχουν βοηθήσει με τον τρόπο τους, και ας μην το γνωρίζουν οι ίδιοι, για να καταφέρω να φτάσω ως εδώ.

Μα πάνω απ' όλα θα ήθελα να ευχαριστήσω την οικογένεια μου. Τους γονείς μου και την αδερφή μου, που μου έχουν προσφέρει τόσα πολλά όλα αυτά τα χρόνια και στους οποίους οφείλω τα πάντα.

Χάρης Αγγελιδάκης

Contents

List of Algorithms

Chapter 1

Introduction

1.1 What is Stochastic Optimization

The study of stochastic optimization dates back to the 1950's and the work of Dantzig [10] and Beale [3], and attempts to model uncertainty in the input data. Uncertainty is a facet of many decision environments and rises naturally in many areas, including transportation models, logistics, financial instruments and network design. In such situations, we have to make decisions in order to satisfy some demands. There are various reasons that cause uncertainty in these demands, such as unpredictable information revealed in the future, or inherent fluctuations caused by noise. In such cases, the demand pattern is not known precisely at the outset, but one might be able to obtain, through simulation models, surveys, market predictions or data-mining techniques, a better understanding and quantification of the nature of future uncertainty with statistical information about the demands. And, while more effective decisions can be made when the actual requirements are given, the decision-making costs are inflated until then. This reflects the increased cost of rapid-response and is the main reason that turns our attention to such problems and makes us try to find ways to make the best possible decisions in advance, and not wait until the exact demands are revealed. Thus, we can say that there is a trade-off between committing initially, having only imprecise information while incurring a lower cost, and deferring decisions to the future when we will have more precise information about the input but the costs will be higher. Stochastic optimization provides a means to handle such uncertainty by modeling it as a probability distribution over the input data.

We will turn our attention to stochastic variants of combinatorial optimization problems. We hope to convince the reader that such problems are worth studying, not only for the sake of theoretical interest, but for that we come across them in many realistic scenarios.

1.2 An example: the 2-stage Stochastic Uncapacitated Facility Location Problem

We will now briefly present a common example of a stochastic problem. Think of a setting where a company has to decide where to set up facilities to serve client demands. Typically the demand pattern is not known precisely at the outset, but one might be able to obtain, as already mentioned, statistical information about the demands. This motivates the following 2-step decision process: in the first stage, given only distributional information about the demands (and deterministic data for the facility opening costs), one must decide which facilities to open initially; once the client demands are realized according to this distribution, we can extend the solution by opening more facilities, incurring a recourse cost, and we have to assign the realized demands to open facilities. This is the *2-stage Stochastic Uncapacitated Facility Location*

Problem. The recourse costs are usually higher than the original ones (because opening a facility later would involve deploying resources with a small lead time); these costs could be different for the different facilities, and could even depend on the realized scenario (i.e. set of demands).

We now proceed to introduce and explain some crucial concepts of stochastic optimization, that will help us define and model our problem formally.

1.3 Modeling our problem

Before continuing, we should make again clear that we are going to focus on combinatorial problems, and thus the solutions we are seeking belong to a finite set that describes all the feasible solutions to our problem. This also means that with the term stochastic optimization, we will almost always imply stochastic combinatorial optimization.

An important concept in stochastic optimization is that of a **scenario**. A scenario is any possible requirement set that can be given as input, and the actual input is the one scenario realized. Thus, the probabilistic information that we have mentioned is actually a probability distribution over the set of scenarios. We assume that the set of scenarios is finite.

Another important concept is that of a **stage**. A stage can be thought of as a discrete moment in time (for example, a single day) in which we obtain some new information about the input. The idea is that there are k such different stages until the exact input is revealed. This is the **k-stage stochastic optimization** problem. When the number of stages is 2, the problem is called *2-stage stochastic.* We refer to problems with 3 or more stages as *multistage stochastic.* We have to point out that in each stage, the costs are different, and usually higher compared to the previous stage, so that it will make sense to take a decision in stage i and not postpone it until the next one. The factor by which the elements at each stage are costlier than the elements in the previous stage is called the **inflation factor**.

The decisions made in each stage are called *recourse actions.* So, we actually name our problems **k-stage stochastic optimization problems with recourse**.

We will now explain what our goal is. As it has already been clear, we consider minimization or maximization problems. And, as we do not know the actual input from the beginning, the best we can do is try to minimize, or maximize, the *expected* total cost incurred. In the following, we will mostly consider minimization problems, but the exact same techniques apply in maximization problems, too.

To make the picture more clear, we will introduce a simple model of an abstract combinatorial minimization problem Π. The optimization problem Π is defined by U, the universe of *clients* (or demands), the set X of *elements* we can purchase, and the cost function $c : X \to \mathbb{R}$. For a subset $F \subseteq X$ of elements, we extend the cost function c so that $c(F) = \sum_{e \in F} c_e$, which is the cost of F. Given a set S of clients, a solution F that *satisfies* each client $j \in S$ is labeled *feasible* for S. The definition of satisfaction is obviously dependent on the problem.

Given a set $S \subseteq U$ of clients, we let $Sols(S) \subseteq 2^X$ be the set of *feasible solutions* for S. So, with client set $S \subseteq U$, the *deterministic version* $Det(\Pi)$ of Π asks us to find a solution $F \in Sols(S)$ of minimum cost. We denote by $OPT(S)$ the cost of this minimum cost solution.

We will now discuss the stochastic variant $Stoc(\Pi)$ of the problem Π, where the set of clients is not known in advance, but is revealed gradually. We proceed to build the solution in stages; in each stage, we gain a more precise estimate of the requirements of clients, and then can buy or extend a partial solution (at gradually increasing cost) in response to this updated information. Ultimately, we learn the entire set S of clients, and then must complete the existing partial solution to a feasible solution $F \in Sols(S)$.

In this *k-stage* problem, we use $\sigma_i^{A_{1\ldots i}}(e)$ to denote the inflation factor of element e in stage i when the "partial" scenario $A_{1\ldots i}$ has been revealed, i.e., how much more expensive each element is in comparison to stage $i-1$, which is a function $\sigma_i^{A_{1\ldots i}} : X \to \mathbb{R}^+$. In many cases the function $\sigma_i^{A_{1\ldots i}}$ is considered constant, so that each element is inflated by the same factor in any scenario. In the more general case, the inflation factor is different for each element and also scenario-dependent, which means that it depends on the outcomes of the previous stages. We also assume that costs are non-decreasing, which corresponds to $\sigma_i^{A_{1\ldots i}}(e) \geq 1$. For completeness, we define $\sigma_1^{A_1}(e) = 1$.

To formalize the way in which information is revealed in each stage, we can think of the client set S as a random variable $\mathbf{S}$ with a probability distribution π, and the information obtained in each stage as random variables $\mathbf{s}_i$ correlated with $\mathbf{S}$. After stage i, we know that future information, as well as the set $\mathbf{S}$ of demands, will come from the conditional distribution $[\pi \,|\, \mathbf{s}_1 = s_1, \mathbf{s}_2 = s_2, ..., \mathbf{s}_i = s_i]$, where $s_1, s_2, ..., s_i$ are the realizations of the random variables $\mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_i$ (note that $\mathbf{s}_1$ is a "dummy" signal and we just use it for convenience). So, when information in stage i has been revealed, we can purchase some more elements $F_i \subseteq X$ at cost $\sum_{e \in F_i} (\prod_{j=1}^{i} \sigma_j^{A_{1\ldots j}}(e)) \cdot c_e$. Finally, in the k^{th} stage, we observe the realization of the random variable $\mathbf{S} = S$, and have to buy the final set F_k so that $\bigcup_{i=1}^{k} F_i \in Sols(S)$. We should note here that throughout this thesis, we assume that the uncertainty is not affected by the decisions taken in previous stages, i.e. the signals $\mathbf{s}_{i+1}, ..., \mathbf{s}_k$ are **conditionally independent** of our decisions $F_1, ..., F_i$. There are problems which bear such a property of dependence (for example, some stochastic scheduling problems) but they are out of the scope of this thesis.

So, what we actually have to do now is minimize the expected cost incurred in all stages together, that is,

$$\sum_{e \in F_1} c_e + \mathbf{E}\Big[\sum_{i=2}^{k} \sum_{e \in F_i} (\prod_{j=1}^{i} \sigma_j^{A_{1\ldots j}}(e)) \cdot c_e\Big]$$

The randomness here is embedded in the sets F_i of each stage $i \geq 2$. Each partial solution set $F_i = F_i(s_1, s_2, ..., s_i)$ may depend on the random variables $\mathbf{s}_1, \mathbf{s}_2, ...\mathbf{s}_i$, but not on any random variable $\mathbf{s}_j, j > i$.

We have to point out here that minimizing (or maximizing) the *expected* value of the objective function is not always desirable, in cases when our decisions produce considerable variations in the actual total costs. In such cases, we may want to take the variance of the random variable into consideration. Such approach is out of the scope of this thesis, and from now on we will care only for the expected value.

The reader should pay attention to the fact that, if we know the whole distribution, the above problem becomes a **fully deterministic** one. However, we will see that it is not really useful to consider problems where the distribution is completely known, as it usually has exponential size. And this is the one thing that is left now to be explained; what we know of the distribution π. Three different models are commonly used to describe the probabilistic part of our problem:

1. *Explicit Scenarios*: In this model we are explicitly given the whole description of the distribution, that is, all the possible realizations of signals in each stage together with their corresponding probabilities. In the special case of 2-stage problems, where it is mostly used, the description is a finite set of scenarios $a_1, a_2, ..., a_k$ which can occur, each with probability $p_1, p_2, ..., p_k$.

2. *Independent Decisions*: In this model, each client j has a probability π_j of arrival *independent* of other clients. Again, it is mostly used in 2-stage problems (although it can also be applied to multistage problems), where the probability $\pi(S)$ of the set S materializing is given by $\prod_{j \in S} \pi_j \cdot \prod_{j \notin S}(1 - \pi_j)$. It can be viewed as a special case of the black-box model presented below. Its main problem is that many of the underlying stochastic applications often involve correlated data (e.g., in stochastic facility location the client demands are expected to be correlated due to economic and/or geographic factors), which this model clearly fails to capture.

3. *Black-Box Model*: This is the most commonly used model, in which we draw samples from a black-box (efficiently), without needing to know the distribution π explicitly. More specifically, in each stage we can sample signals of the following stages in order to get a picture of how the distribution is approximately. It is the most general model, as it can succinctly describe arbitrary distributions with exponential size support. Note that the running time of an algorithm in the black-box model depends on the number of samples drawn, as each call to the black-box is considered an elementary operation.

We are going to use solely the black-box model, which is the most general and widely used one. However, some references to the polynomial-scenario model (that is, the "explicit scenarios" with a polynomial number of scenarios) will be made, as some results concerning that model will be utilized in order to reach approximation algorithms for 2-stage problems in the black-box model. More details of this will be presented in chapter 5.

1.4 What can be found in this thesis - Organization of material

In this thesis, we have tried to cover a part of the research done in Stochastic Combinatorial Optimization the last decade. We have focused on the black-box model, as it is the most powerful one and it seems that it can model most problems encountered in practical applications.

The text is divided in two parts, the first dealing with 2-stage problems, while the second dealing with multistage problems. Although our goal was to make these two parts readable independently, it is strongly suggested that the unfamiliar reader begins with the first part, as 2-stage problems introduce most concepts of stochastic optimization in a much easier and clear way. As regards the chapters of each part, these can definitely be studied out of order, although we believe that the chosen order is the most adequate to build up in the ideas and techniques presented.

Our purpose was to make this thesis as self-contained as possible, so that a reader who is unfamiliar with the field can get a good grasp of it after reading it. Of course, since we deal with approximation algorithms for variants of combinatorial problems that involve probabilistic input, some basic familiarity with approximation algorithms, randomized algorithms and classic combinatorial optimization techniques, such as LP formulations, relaxations and roundings, is assumed. We will also need some notions and techniques from linear algebra and convex optimization. For this, we have included two short appendices, which, although should in no case be considered complete, we suggest the reader to go through them. In any case, we hope that the references given will make amends for any vague parts of the text.

Part I

2-stage Stochastic Optimization

Chapter 2

The 2-stage Stochastic Optimization Problem

In the following chapters we will focus on algorithms for solving 2-stage stochastic variants of combinatorial (minimization) problems. The 2-stage stochastic model with recourse is the most widely used model in stochastic optimization and has been extensively studied the last decade. We will restate the problem in its most famous paraphrase:

> "On Monday, we only know the input distribution on the clients, and we can buy some resources. On Tuesday, the client set is completely specified, but things are more expensive. What we have to do now is buy any additional resources needed to get a feasible solution to the instance."

More formally, in the first stage we are given a probability distribution over possible scenarios and we construct an anticipatory part of the solution, x, incurring a cost $c(x)$. Then, in the second stage, a scenario A materializes according to the distribution, and we may augment the initial decisions x by taking recourse actions y_A, if necessary, incurring a certain cost $f_A(x, y_A)$, such that $x \cup y_A$ is a feasible solution for scenario A. Our goal is to choose the initial decisions so as to minimize the *expected* total cost,

$$c(x) + \mathbf{E}_A[f_A(x, y_A)],$$

where the expectation is taken over all scenarios according to the given probability distribution.

Observe that in order to minimize the above expected total cost, when a particular scenario A realizes and given a first-stage action x, we actually have to solve the deterministic minimization problem

$$\min_{y \in R}\{f_A(x, y) \mid (x, y) \text{ is a feasible solution for scenario } A\},$$

where R is the set of recourse actions.

As already mentioned, one of the main issues that arise in such problems is how the scenario distribution is represented. The simplest approach is to assume that we are given, as part of the input, a list that explicitly enumerates each scenario and its probability of occurrence. However, this causes a significant blow-up in the input size, as in most cases the distribution has support of size exponential in the other input parameters, as it will be observed in the problems we will discuss. Thus, to ensure that a polynomial time algorithm in this model has running time polynomial in all input parameters, one must restrict oneself to distributions with a polynomial-size support, which is a severe restriction.

One solution is to use the independent-decisions model (also known as independent-activation model), where the scenario distribution is a product of independent distributions that are described explicitly in the input. This model allows one to succinctly specify a class of distributions with exponentially many scenarios. However, it fails to capture situations where data are correlated, as already mentioned.

Thus, from now on we are going to use the black-box model, which is a more general way of describing the distribution. In the black-box model one can use a procedure to *sample* scenarios from the distribution. Each call to this procedure is treated as an elementary operation and the running time of an algorithm is measured in terms of the number of such calls. The black-box model incorporates the desirable aspects of both the previous models: it allows one to specify distributions with exponentially many scenarios and correlation in a compact way that makes it reasonable to talk about polynomial time algorithms. Note that the realization of a scenario in the second stage is probabilistically equivalent to drawing an independent sample from the black-box. This observation will prove useful.

We believe that the reader has reached the point where he can guess that stochastic optimization problems are generally hard. In most cases, the computational difficulty stems from the fact that the number of scenarios is exponential, a phenomenon often call the "curse of dimensionality". This, combined with the fact that the problems we are going to discuss are NP-*hard* even in their deterministic counterparts, makes it clear that we are going to focus on approximation and not exact algorithms.

More specifically, since the black-box model does not allow us to know the whole input, it is obvious that any algorithm discussed will be randomized, in the sense that there can be no deterministic algorithm when access to the input is only partial through sampling. So, ideally, a "good" algorithm for a 2-stage stochastic problem must suggest a way to determine the first-stage decisions with the least possible calls to the black-box so as to keep the expected total cost, that is the cost of the first-stage and the expected cost of the second stage, as low as possible. Once a particular scenario is realized, it should find an augmenting solution of minimum cost. Note that the second-stage solutions need not be precomputed, and this actually is impossible when we use the black-box model. The guarantee provided is in any case probabilistic, so we should keep in mind that we are looking for algorithms that work well with high probability.

Chapter 3

The Boosted Sampling Framework

3.1 Introduction

We are going to start our overview of algorithms for 2-stage problems by presenting a general technique, the *Boosted Sampling* framework, to adapt approximation algorithms for several deterministic combinatorial problems to their 2-stage stochastic versions. This framework was introduced by Gupta, Pál, Ravi and Sinha in [17], and it was the first work that dealt with 2-stage problems in the black-box model. Until then, research had been done only for the "explicit scenarios" and "independent decisions" models. It provides a simple method that can give algorithms with good performance guarantees (on expectation) for problems that satisfy some restrictions. In its initial version, three restrictions are imposed: (a) the problems must satisfy sub-additivity, (b) the inflation factor is constant, and (c) the deterministic analog has an α-approximation algorithm $\mathcal{A}$ that admits a β-strict cost-sharing function. We will explain all these in detail.

An extension of the framework for dealing with multistage stochastic optimization problems, which will be presented in chapter 8, as well as for handling inflation factors arbitrarily correlated with scenarios, was given one year later by Gupta etc. [18]. It is an open question whether the framework can be extended to handle multiple inflation factors, that is, different inflation factors for different elements in each scenario.

In this chapter, we will begin by analyzing the initial, simpler version of the framework where all three restrictions are imposed, and then we will show how we can make some changes in order to handle cases where the inflation factor is not constant. We will also see how this framework can be applied to give the first constant-factor approximation algorithms for 2-stage Stochastic Set Cover, Vertex Cover and Facility Location Problems in the black-box model.

3.2 Model

We are going to use the definition of an abstract deterministic combinatorial optimization problem Π, as given in section 1.3. The 2-stage stochastic variant is now defined as follows:

- There will be *two stages* of purchasing. Let $\sigma \geq 1$ be a given inflation parameter; every element $x \in X$ costs c_x in the first stage and σc_x in the second.
- In the first stage, we are given access to a *black-box* from which we can draw samples from the probability distribution $\pi : 2^U \to [0,1]$ in time $poly(|U|)$. We can then construct a *first-stage solution* by buying a set of elements F_0 at cost $c(F_0)$.
- In the second stage, one set $S \subseteq U$ of clients is realized according to the distribution π (the probability that S is realized is $\pi(S)$). We remind the reader of the assumption that

this set S is *conditionally independent* of our actions in the first stage. Now, the second-stage solution (or, the *recourse* solution) consists of a set F_S purchased at the inflated cost $\sigma c(F_S)$, so that $F_0 \cup F_S \in Sols(S)$.

Our objective is to select a set F_0 and then, given a set S from the distribution π, to select the set F_S so as to minimize the expected cost of the solution, that is:

$$\text{min:} \quad c(F_0) + \sum_{S \subseteq U} \pi(S)\sigma c(F_S)$$

We are now going to explain the restrictions we mentioned above that we impose on our problems (more precisely, on the deterministic versions of our problems).

3.2.1 Sub-additivity

Definition 3.1. Let Π be a combinatorial optimization problem, as already defined. The problem Π is sub-additive, if for any sets S and S' of clients with solutions $F \in Sols(S)$ and $F' \in Sols(S')$, (i) the set $S \cup S'$ is a legal set of clients for Π, and (ii) $F \cup F' \in Sols(S \cup S')$.

For example, the *Steiner Tree Problem* in its general form is not sub-additive, as the set $F \cup F'$ may be a forest and not a tree. To ensure sub-additivity, we require a root vertex r. So, given a set S, we can view it as trying to solve the classic Steiner Tree Problem for the set $S \cup \{r\}$. This is the *Rooted Steiner Tree Problem.*

Sub-additivity expresses in a way the economies of scale. It simply means that the cost of constructing a solution for $S \cup S'$ is less than or equal to the cost of constructing solutions for S and S' independently, as the two independent solutions are also solutions for $S \cup S'$.

3.2.2 Cost-sharing functions

Definition 3.2. Let $\chi : 2^U \times U \to \mathbb{R}_{\geq 0}$ be a function that, for every demand set $S \subseteq U$ of our problem Π, assigns a non-negative real value $\chi(S, e)$ to each client $e \in S$. The function is then called a *cost-sharing method.* For convenience, we also define $\chi(S, e) = 0$, $\forall e \notin S$.

Loosely, a cost-sharing function is one that divides the cost of a solution $F \in Sols(S)$ among the client set S. Cost-sharing functions have long been used in the context of game-theory. We will use a variant of the above definition and define our cost-sharing functions relative to an approximation algorithm $\mathcal{A}$ for the problem Π.

Definition 3.3. A cost-sharing algorithm $\mathcal{A}$ for a problem Π takes an instance (X, S) of Π and outputs

1. a solution $F \subseteq X$ with $F \in Sols(S)$, and
2. a real value $\xi(X, S, j) \geq 0$ for each client $j \in S$ (and $\xi(X, S, j) = 0$ for $j \notin S$).

The value $\xi(X, S, j)$ is called the cost-share of client j, and the function $\xi(\cdot, \cdot, \cdot)$ computed by $\mathcal{A}$ is the cost-sharing function associated with $\mathcal{A}$.

A useful property of cost-sharing functions is to provide a lower bound on the cost of the optimal solution, so that we can use the cost-sharing function to provide bounds on the cost of our solution.

Definition 3.4 (Competitiveness)**.** A cost-sharing function ξ is competitive if for every client set S, it holds that

$$\sum_{j \in S} \xi(X, S, j) \leq OPT(X, S)$$

Note 3.1. *From now on, the notation $OPT(X, S)$ will appear instead of $OPT(S)$ to denote the cost of the optimal solution for the demand set S, when we want to explicitly state the set of elements X from which we construct our solution. Also, we define the function $\xi(X, S, A)$ as the sum $\sum_{j \in A} \xi(X, S, j)$.*

Intuitively, competitiveness tells us that dividing the cost of a solution among the client set is done in such a way so as not to "overcharge" any demand j, and, as a result, not make "overcharges" in total.

Another property, crucial to our analysis, is strictness, which relates the cost of extending a solution on S so as to serve more clients T to the cost shares of T. Formally, given a set X of elements and S of clients, let $\mathcal{A}(X, S)$ denote the solution found by algorithm $\mathcal{A}$.

Definition 3.5 (Strictness)**.** A cost-sharing algorithm $\mathcal{A}$ is β-strict if for any sets of clients S, T, there exists a solution $F_T \subseteq X$ constructible in polynomial time such that $\mathcal{A}(X, S) \cup F_T \in Sols(T)$ and $c(F_T) \leq \beta \cdot \xi(X, S \cup T, T)$.

What we actually require with strictness is the existence of a polynomial time algorithm $Aug_{\mathcal{A}}$ which can augment $\mathcal{A}(X, S)$ to a solution in $Sols(T)$ at cost at most $\beta \cdot \xi(X, S \cup T, T)$. Observe that if the problem is sub-additive, then we also have that $\mathcal{A}(X, S) \cup F_T \in Sols(S \cup T)$. Sub-additivity also gives the stricter inequality $c(F_T) \leq \beta \cdot \xi(X, S \cup T, T \setminus S)$, since augmenting a solution $\mathcal{A}(X, S)$ to a solution in $Sols(T \setminus S)$ gives a solution in $Sols(S \cup (T \setminus S)) = Sols(S \cup T)$, and, thus, it is also a solution in $Sols(T)$.

Intuitively, the β-strictness ensures that extending an existing solution to satisfy new demands will not cost much more than knowing the whole set of demands from the beginning. More precisely, we only pay a factor of β as far as the new clients are concerned, using the augmenting algorithm.

The reader should also have in mind that in all known cases, approximation algorithms with β-strict cost-sharing functions are obtained via the primal-dual scheme, and the cost-shares are derived from the dual variables. Thus, boosted sampling can be viewed as a primal-dual approach for designing approximation algorithms for stochastic problems.

Note 3.2. *One obvious algorithm Aug_A can be obtained by just zeroing out the costs of elements already picked in $\mathcal{A}(X, S)$, and running $\mathcal{A}$ again.*

For the needs of this chapter, when we say that there is a cost-sharing function ξ that is strict w.r.t. an approximation algorithm $\mathcal{A}$, we mean that there is a competitive and β-strict cost-sharing function ξ associated with $\mathcal{A}$.

3.3 The Boosted Sampling Algorithm

3.3.1 The initial Boosted Sampling

We are now going to present the algorithm. Three restrictions, as already mentioned, are imposed on the problem Π: sub-additivity, constant inflation factor, and that there is a cost-sharing function ξ that is strict w.r.t. $\mathcal{A}$. Given an instance of a stochastic problem $Stoc(\Pi)$, the goal of the first stage is to buy the elements that will be useful for the unknown client set realized in the second stage. As we cannot see the future, the next best thing to do is to sample from the distribution π (through the black-box), and use the samples as an indication of what the future set will be. This very simple idea is the basis of this method.

A first attempt could be to sample once from the distribution and use the set obtained as our prediction. However, this approach ignores the fact that the future is more expensive by a factor of σ, and in cases when σ is large, our solution would not be good. In fact, we can see that, as $\sigma \to \infty$, the optimal solution would be to assume that every client in U will be realized and must be accounted for in the first stage itself. We will see that in the end a number of $\lfloor\sigma\rfloor$ samples is adequate. Intuitively, this tries to account for the σ inflation factor by sampling each scenario A, in expectation, $\sigma \cdot p_A$ times.

Algorithm 1: `Boost-and-Sample`(Π)

```
1. Draw $\lfloor\sigma\rfloor$ independent samples $D_1, D_2, ..., D_{\lfloor\sigma\rfloor}$ by sampling from the
   distribution $\pi$. Let $D = \bigcup_i D_i$.

2. Using the algorithm $\mathcal{A}$, construct an $\alpha$-approximate first-stage solution
   $F_0 \in Sols(D)$.

3. If the client set $S$ is realized in the second stage, use the augmenting
   algorithm $Aug_{\mathcal{A}}$ to compute $F_S$ such that $F_0 \cup F_S \in Sols(S)$.
```

Theorem 3.1. *Consider a combinatorial optimization problem Π that is sub-additive, and let $\mathcal{A}$ be an α-approximation algorithm for its deterministic version $Det(\Pi)$ that admits a β-strict cost-sharing function. Then, the algorithm* `Boost-and-Sample`(Π) *presented above is an $(\alpha+\beta)$-approximation algorithm for $Stoc(\Pi)$.*

Note 3.3. *The approximation factor in the above algorithm concerns the expected value that the algorithm returns, as there is some randomness involved in the first step of the algorithm, in which we sample from our distribution. In any case, the solution returned is a feasible one for the problem.*

Proof. We are going to bound the expected costs of our first and second-stage solutions separately. Let F_0^* be the first-stage component of the optimal solution, and F_S^* be the second-stage component if the set realized is S. Hence the optimal cost is:

$$Z^* = c(F_0^*) + \sigma \sum_S \pi(S)c(F_S^*)$$

First stage: The first-stage solution we use is an α-approximate solution for the demand set D. Let $F_{D_1}^*, F_{D_2}^*, ..., F_{D_{\lfloor\sigma\rfloor}}^*$ be the optimal second-stage solutions when sets $D_1, D_2, ..., D_{\lfloor\sigma\rfloor}$ realize, respectively. We have that $F_0^* \cup F_{D_1}^* \in Sols(D_1)$, $F_0^* \cup F_{D_2}^* \in Sols(D_2)$, ..., $F_0^* \cup F_{D_{\lfloor\sigma\rfloor}}^* \in Sols(D_{\lfloor\sigma\rfloor})$, and as the problem Π is sub-additive and $D = \bigcup_i D_i$, we get that $\widehat{F}_1 = F_0^* \cup F_{D_1}^* \cup F_{D_2}^* \cup ... \cup F_{D_{\lfloor\sigma\rfloor}}^* \in Sols(D)$.

The cost of the solution $\widehat{F}_1$ is $c(\widehat{F}_1) \le c(F_0^*) + \sum_{i=1}^{\lfloor\sigma\rfloor} c(F_{D_i}^*)$, and so we get that:

$$\begin{aligned}
\mathbf{E}[c(\widehat{F}_1)] &\le \mathbf{E}[c(F_0^*)] + \sum_{i=1}^{\lfloor\sigma\rfloor} \mathbf{E}[c(F_{D_i}^*)] \\
&= c(F_0^*) + \lfloor\sigma\rfloor \sum_S \pi(S)c(F_S^*) \\
&\le c(F_0^*) + \sigma \sum_S \pi(S)c(F_S^*) \\
&= Z^*
\end{aligned}$$

(Each D_i is chosen independently from the probability distribution π, and this is why we have $\mathbf{E}[c(F_{D_i}^*)] = \sum_S \pi(S)c(F_S^*)$ for each i.)

Since we use an α-approximation algorithm, our solution F_0 satisfies $c(F_0) \leq \alpha c(\widehat{F}_1)$, because, as we have already mentioned, $\widehat{F}_1 \in Sols(D)$, and so we get $\mathbf{E}[c(F_0)] \leq \alpha Z^*$.

Second stage: Let S be the set of realized clients, and let F_S be the result of our algorithm $Aug_{\mathcal{A}}$ such that $F_0 \cup F_S \in Sols(S)$. We are going to bound our expected second stage cost, which is $\sigma\, \mathbf{E}[c(F_S)]$. From the β-strictness of the cost-sharing function and the sub-additivity of our problems we get that $c(F_S) \leq \beta\, \xi(X, D \cup S, S \setminus D)$, which gives

$$\mathbf{E}_S[c(F_S)] \leq \beta\, \mathbf{E}_{D,S}[\xi(X, D \cup S, S \setminus D)]. \tag{3.1}$$

In order to bound the expectation in the right hand side of the above inequality, we are going to consider an alternate probabilistic process to generate the sets D_i and the set S, which is identically distributed to the original one. We draw $\lfloor\sigma\rfloor + 1$ independent samples $\widehat{D}_1, \widehat{D}_2, ..., \widehat{D}_{\lfloor\sigma\rfloor+1}$ from the distribution π. We now choose a random value K from $1, 2, ..., \lfloor\sigma\rfloor + 1$, and set $S = \widehat{D}_K$ and $D = \bigcup_{i \neq K} \widehat{D}_i$. This process is indeed identically distributed to the original one, as we pick the sets independently (remember that realization of a scenario is identical to drawing an independent sample from the black-box). Let $\widehat{\mathbb{D}}$ be the union of all the $\widehat{D}_i$'s, and let $\widehat{\mathbb{D}}_{-i}$ be the union $\bigcup_{l \neq i} \widehat{D}_l$ of all the sets except $\widehat{D}_i$.

From the above definition, we can easily see that $(\widehat{D}_i \setminus \widehat{\mathbb{D}}_{-i}) \cap (\widehat{D}_j \setminus \widehat{\mathbb{D}}_{-j}) = \emptyset$ for $i \neq j$, and $\bigcup_i (\widehat{D}_i \setminus \widehat{\mathbb{D}}_{-i}) = \widehat{\mathbb{D}}$. Thus, using the competitiveness of the cost-sharing function we get that

$$\begin{aligned}
&\xi(X, \widehat{\mathbb{D}}, \widehat{\mathbb{D}}) \leq OPT(\widehat{\mathbb{D}}) \Rightarrow \\
&\sum_{i=1}^{\lfloor\sigma\rfloor+1} \xi(X, \widehat{\mathbb{D}}, \widehat{D}_i \setminus \widehat{\mathbb{D}}_{-i}) \leq OPT(\widehat{\mathbb{D}}).
\end{aligned}$$

By our random choice of K, we get

$$\begin{aligned}
\mathbf{E}_K[\xi(X, \widehat{\mathbb{D}}, \widehat{D}_K \setminus \widehat{\mathbb{D}}_{-K})] &= \frac{1}{\lfloor\sigma\rfloor + 1} \sum_{i=1}^{\lfloor\sigma\rfloor+1} \xi(X, \widehat{\mathbb{D}}, \widehat{D}_i \setminus \widehat{\mathbb{D}}_{-i}) \\
&\leq \frac{1}{\lfloor\sigma\rfloor + 1} OPT(\widehat{\mathbb{D}})
\end{aligned}$$

Since the alternate process is probabilistically identical, as we have already mentioned, to the one we used to pick D and S in our algorithm, we have

$$\begin{aligned}
\mathbf{E}_{D,S}[\xi(X, D \cup S, S \setminus D)] &= \mathbf{E}_{\widehat{\mathbb{D}},K}[\xi(X, \widehat{\mathbb{D}}, \widehat{D}_K \setminus \widehat{\mathbb{D}}_{-K})] \\
&\leq \frac{1}{\lfloor\sigma\rfloor + 1} \mathbf{E}_{\widehat{\mathbb{D}}}[OPT(\widehat{\mathbb{D}})]
\end{aligned} \tag{3.2}$$

The last thing left to do now is bound $\mathbf{E}_{\widehat{\mathbb{D}}}[OPT(\widehat{\mathbb{D}})]$. A feasible solution to $\widehat{\mathbb{D}}$ is $\widehat{F}_2 = F_0^* \cup F^*_{\widehat{D}_1} \cup F^*_{\widehat{D}_2} \cup ... \cup F^*_{\widehat{D}_{\lfloor\sigma\rfloor+1}}$. Again, like the solution $\widehat{F}_1$ defined in the first stage, $\widehat{F}_2 \in Sols(\widehat{\mathbb{D}})$

because Π is sub-additive. Thus, we have:

$$\begin{aligned}
OPT(\widehat{\mathbb{D}}) \le c(\widehat{F}_2) &\le c(F_0^*) + \sum_{i=1}^{\lfloor\sigma\rfloor+1} c(F^*_{\widehat{D}_i}) \Rightarrow \\
\mathbf{E}_{\widehat{\mathbb{D}}}[OPT(\widehat{\mathbb{D}})] &\le c(F_0^*) + \mathbf{E}_{\widehat{\mathbb{D}}}[\sum_{i=1}^{\lfloor\sigma\rfloor+1} c(F^*_{\widehat{D}_i})] \\
&= c(F_0^*) + \sum_{i=1}^{\lfloor\sigma\rfloor+1} \mathbf{E}_{\widehat{D}_i}[c(F^*_{\widehat{D}_i})] \\
&= c(F_0^*) + (\lfloor\sigma\rfloor + 1)\sum_S \pi(S)c(F_S^*) \\
&= c(F_0^*) + \frac{\lfloor\sigma\rfloor + 1}{\sigma}\sigma\sum_S \pi(S)c(F_S^*) \\
&\le \frac{\lfloor\sigma\rfloor + 1}{\sigma}\Big(c(F_0^*) + \sigma\sum_S \pi(S)c(F_S^*)\Big) \\
&= \frac{\lfloor\sigma\rfloor + 1}{\sigma}\, Z^*
\end{aligned}$$

Using this last inequality and (3.1) and (3.2) we get

$$\begin{aligned}
\mathbf{E}_S[c(F_S)] &\le \beta\,\frac{1}{\lfloor\sigma\rfloor + 1}\,\frac{\lfloor\sigma\rfloor + 1}{\sigma}\, Z^* \\
&= \frac{\beta}{\sigma}\, Z^*.
\end{aligned}$$

Thus, the expected second stage cost is $\sigma\mathbf{E}_S[c(F_S)] \le \beta Z^*$, and so we get a bound of $(\alpha+\beta)$ for the total expected cost:

$$\mathbf{E}[c(F_0) + \sigma c(F_S)] \le (\alpha + \beta)\, Z^*$$

This completes our proof.

□

Note 3.4. *The running time of the algorithm is polynomial in the input and in the inflation factor σ. In most cases, it is reasonable to assume that σ is polynomial in the size of the deterministic version of the problem, and can be considered a small number. In such cases, the above algorithm is polynomial. However, when σ is very large, for example exponentially large compared to a parameter n of the input, then the algorithm becomes exponential.*

3.3.2 Correlated Inflation Factors: extending the Boosted Sampling framework

We will now show how we can extend the basic Boosted Sampling framework in order to handle cases where the inflation factor σ is a random variable arbitrarily correlated with the random scenarios. This extension was given in [18].

Formally, let us assume that we have access to a distribution π' over $\mathbb{R}_{\ge 1}\times 2^U$, where $\pi'(\sigma, S)$ is the probability that the set S arrives and the inflation factor is σ. We assume that we know an integer $M \in \mathbb{Z}$ which is an upper bound on the value of the inflation parameter σ, i.e., with probability 1, it should be the case that $\sigma \le M$ holds. We will see that in most algorithms presented, this upper bound on the inflation parameter is required. However, we believe that it is a sensible assumption for most realistic problems. Note also that choosing a pessimistic value of M will only increase the running time, but not degrade the approximation guarantee of the framework.

The modified algorithm is now:

Algorithm 2: `General-Boost-and-Sample`(Π)

```
1. Draw $M$ independent samples from the joint distribution $\pi'$ of $(\boldsymbol{\sigma}, \mathbf{S})$. Let
   $(\sigma_1, S_1), ..., (\sigma_M, S_M)$ denote this collection of samples.

2. For $i = 1, ..., M$, accept the sample $S_i$ with probability $\sigma_i/M$. Let $S_{i_1}, ..., S_{i_k}$
   be the accepted samples, and let $S = \bigcup S_{i_j}$.

3. Using the algorithm $\mathcal{A}$, construct an $\alpha$-approximate first-stage solution
   $F_0 \in Sols(S)$.

4. If the client set $T$ is realized in the second stage, use the augmenting
   algorithm $Aug_{\mathcal{A}}$ to compute $F_T$ such that $F_0 \cup F_T \in Sols(T)$.
```

Note that if σ is constant, then we get the original Boosted Sampling framework. To get some intuition about the changes, observe that if the sampled inflation factor σ_i is large, this indicates that we want to handle the associated S_i in the first stage; on the other hand, if the σ_i is small, we can afford to wait until the second-stage to handle the associated S_i and this is indeed what the algorithm does, albeit in a probabilistic way.

Theorem 3.2. *Consider a sub-additive combinatorial optimization problem Π, and let $\mathcal{A}$ be an α-approximation algorithm for its deterministic version Det(Π). If $\mathcal{A}$ admits a β-strict cost-sharing function, then* `General-Boost-and-Sample`(Π) *is an $(\alpha + \beta)$-approximation algorithm for Stoc(Π).*

Outline of the Proof. The main idea is to transform the random inflation stochastic problem instance (X, π') to one with a fixed inflation factor, and show that the original Boosted Sampling framework runs the same way in this new instance as the General Boosted Sampling framework in the initial instance (X, π'). For this reason, we define the distribution:

$$\widehat{\pi}(\sigma, S) = \pi'(\sigma, S) \times (\sigma/M).$$

We have that $\sum_{\sigma,S} \widehat{\pi}(\sigma, S) \leq 1$, and in order to have a well-defined probability distribution, we increase the probability $\widehat{\pi}(1, \emptyset)$ so that the above sum becomes exactly 1. The inflation factor for this new instance is set to M, and hence the σ output by $\widehat{\pi}$ is only for expositional ease. Note that the probability of a particular set of demands S now is $\sum_{\sigma}(\pi'(\sigma, S) \cdot \sigma/M)$, as we take into account situations where we have instances (σ_1, S), (σ_2, S) with $\sigma_1 \neq \sigma_2$.

The objective now for this new instance is to minimize the expected cost under this new distribution, which is

$$c(F_0) + \sum_{\sigma,S} \widehat{\pi}(\sigma, S) M c(F_S).$$

This is equal to $c(F_0) + \sum_{\sigma,S} \pi'(\sigma, S)(\sigma/M) M c(F_S) = c(F_0) + \sum_{\sigma,S} \pi'(\sigma, S) \sigma c(F_S)$, and, so, our transformed problem objective function is identical to the original objective function. Thus, the two problems are identical, and running `Boost-and-Sample` on this new distribution $\widehat{\pi}$ with inflation parameter M would give us an $(\alpha + \beta)$-approximation.

However, in order to run the initial algorithm in the new distribution, this new distribution must be "implemented" in some way. And this is exactly what `General-Boost-and-Sample` does. We can implement $\widehat{\pi}$ given black-box access to π' by just rejecting any sample (σ, S) with probability σ/M. Note that the rejected samples can be viewed as realizations of the null $(\emptyset)$ scenario (we remind the reader here that we have increased the probability of the null scenario in order for $\widehat{\pi}$ to be well-defined, and these empty samples occur with this exact probability), and so, we actually have M samples, $M-k$ of which are the empty set. With this in mind, the first-part of `Boost-and-Sample` can run in this instance exactly as `General-Boost-and-Sample` does.

The only thing left to comment on regards the realization of a scenario in the second stage. The scenario that realizes in any case comes from the initial distribution π'. So, one could say that we run into a problem here. However, we note that, while in $\widehat{\pi}$ each scenario has different probability of occurrence compared to π', the inflation factor that `General-Boost-and-Sample` uses in any case is the inflation factor that realizes in the second stage according to π'. So, while the initial `Boost-and-Sample` would use M as the inflation factor, by using the σ that realizes we actually succeed in tweaking the distribution π', so as to behave exactly as $\widehat{\pi}$ would in the initial framework. The unconvinced reader can take some time and do the math in order to see that this is indeed the case.

□

3.4 Applications

In this section, we will present some applications of the Boosted Sampling framework to problems that satisfy the restrictions needed. We will not go into any technical details regarding individual problems.

3.4.1 The Rooted Steiner Tree Problem

Theorem 3.3. *There exists a 2-approximation algorithm for the Steiner Tree Problem that admits an 2-strict cost-sharing function.*

The 2-approximation algorithm is simply Prim's algorithm (which can be viewed as a primal-dual algorithm) for finding a Minimum Spanning Tree (MST) over the required vertices plus the root. Now, combining this with Boosted Sampling, we get the following theorem.

Theorem 3.4. *There exists a 4-approximation algorithm for the 2-stage Stochastic Rooted Steiner Tree Problem.*

3.4.2 The Metric Uncapacitated Facility Location Problem

Before mentioning the results that we have for the problem, let us state the problem itself. In the deterministic Metric Uncapacitated Facility Location (DMUFL) Problem, given a set of candidate facility locations $\mathcal{F}$ and a set of clients $\mathcal{D}$, we want to open facilities at a subset of the locations in $\mathcal{F}$, and assign each client to an open facility. Opening a facility at location i incurs a cost of f_i , and the cost of assigning client j to facility i is $d_j \cdot c_{ij}$ where d_j is the demand of client j, c_{ij} is the distance between i and j, and the distances c_{ij} form a metric. The goal is to minimize the total facility opening costs and client assignment costs.

In the deterministic problem one assumes that the client demands are precisely known in advance; the 2-stage Stochastic Uncapacitated Facility Location (SUFL) Problem handles settings where there is uncertainty in the demand, for example, due to macro-economic factors such as competition, technology, or customer purchasing power. We are given a probability distribution on tuples $(d1, ..., d_{|\mathcal{D}|})$ where $d_j \in \{0, 1, ..., D\}$ specifies the demand of client j and D is some known upper bound on the demand. We can open some facilities in stage I paying a cost of $f_i^{(1)}$ for opening facility i, then the actual scenario A with demands d_j^A is revealed, and we may choose to open some more facilities in stage II, incurring a cost of σf_i for each facility i that we open in scenario A.

Theorem 3.5. *The cost-sharing function given by Pál and Tardos [34] is 5.45-strict for the 3-approximation algorithm of Mettu and Plaxton [30]. Hence, there is a 8.45-approximation algorithm for the 2-stage Stochastic MUFLP.*

3.4.3 Vertex Cover

The Boosted Sampling Algorithm can also be used to give a constant-approximation for the 2-stage Stochastic Vertex Cover. In the 2-stage Stochastic Vertex Cover the edges correspond to the demands, and the uncertainty is over which edges will actually be in our graph in the second stage. The algorithm used is a standard primal-dual algorithm which gives cost-shares that are 6-strict with respect to it. Thus, we get the following result.

Theorem 3.6. *There exists a 2-approximation algorithm for Vertex Cover that admits a 6-strict cost-sharing function. Thus, we get an 8-approximation algorithm for the 2-stage Stochastic Vertex Cover.*

Chapter 4

2-stage Stochastic LP's and rounding techniques

4.1 Introduction

In this chapter we are going to formulate our problem as an Integer Program, and then try to find rounding techniques that can guarantee that a solution to the corresponding linear relaxation could give a satisfactory solution to the initial problem. LP rounding is a very common technique that gives approximation algorithms for many deterministic combinatorial optimization problems. The reader can refer to [47] for more details.

Two recent works on two-stage stochastic optimization base their results on rounding fractional solutions of LP's, the first being the work of Gupta, Ravi and Sinha [19] and the second the work of Shmoys and Swamy [38]. The first considers problems with a polynomial number of scenarios, as well as risk-aversion problems (we will talk about them in the final chapter of this thesis), and gives constant-factor approximation algorithms for the 2-stage Stochastic Steiner Tree, for both risk-bounded and not risk-bounded variants.

We will focus on the work of Shmoys and Swamy and present their rounding scheme and the ellipsoid-based algorithm they suggest to solve the linear relaxation of such problems. We will work in the black-box model with arbitrary probability distributions and any number of scenarios. More specifically, we will show that for a class of set cover instances there exist good rounding techniques that can give an approximation guarantee. This motivates us to try and solve the linear relaxation of such problems. Of course, such LP's are not easy to solve. However, we will manage to reach a Fully Polynomial Randomized Approximation Scheme (FPRAS) for a wide class of 2-stage stochastic linear programs by utilizing the ellipsoid method. We will mostly deal with the set cover problem, and at the end of the chapter we will generalize to a wider class of stochastic programs. The main difference of this approach compared to the boosted sampling framework of chapter 3 is that, although cost-shares are not required, suitable LP-relaxation is needed with the ability to round each stage independently.

Before continuing to our analysis, we should mention that the results reached in this chapter can also be obtained by utilizing the Sample Average Approximation Method (SAA), and we will actually present this method in the following chapters. The SAA method is a more natural and simple approach to such problems and can be shown to work well. In practice, it is much more preferable and better implemented than the ellipsoid method. Swamy and Shmoys ([43]) use this method to obtain approximation algorithms for multistage stochastic problems, and as a special case, they reach the same results that they obtain through the ellipsoid method for two-stage stochastic problems, as their approach in both cases is based on the subgradients of the objective value functions. Their SAA approach for two-stage problems can also be found in

[42].

Another approach through the SAA method for two-stage problems can be found in [7], and we will actually present this SAA approach in chapter 5. The reader can already guess that the SAA is much more elegant and avoids the machinery of the ellipsoid algorithm. However, we believe that both methods are worth studying, as each one reveals different aspects of the problems in discussion and provides diverse tools in our attempt to solve them efficiently.

4.2 The Stochastic Set Cover

4.2.1 An initial approach

The deterministic weighted Set Cover Problem (DSC) is the following: given a universe $U = \{e_1, e_2, ..., e_n\}$ of elements and a collection of subsets of U, $\mathcal{S} = \{S_1, S_2, ..., S_m\}$, with set $S_i \subseteq U$ having weight w_i, we want to choose a minimum-weight collection of sets so that every element e_j, $j = 1, 2, ..., n$, is included in some chosen set. The problem can be formulated as an integer program with decision variables $x_1, x_2, ..., x_m$, each one corresponding to a set S_i and having value 1 if this set is chosen, otherwise having value 0, and the integrality constraints can be relaxed to yield the following linear program:

(SC-P):

$$\begin{aligned} \text{min:} \quad & \sum_{S \in \mathcal{S}} w_S x_S & \\ \text{subject to:} \quad & \sum_{S \in \mathcal{S}: e \in S} x_S \geq 1 & \text{for all } e \\ & x_S \geq 0 & \text{for all } S \end{aligned}$$

In the 2-stage stochastic variant of the problem, the elements to be covered are not known in advance. All we know is a probability distribution over scenarios, and each scenario specifies the actual set of elements $A \subseteq U$ to be covered. A scenario is just a subset of the elements. Thus, the set of all possible scenarios (including the empty set) is the power set 2^U. The probability of scenario A is denoted p_A.

Each set S_i has a first-stage weight $w_{S_i}^{(1)}$ and a second-stage weight $w_{S_i}^{(2)} \geq w_{S_i}^{(1)}$ (it will be shown in section 4.3 that the second stage costs can be scenario-dependent. However, for now, we consider them independent of the distribution). In the first stage, we select some of these sets. Then a scenario A realizes according to the distribution, and then additional sets may be selected so as to ensure that A is covered in the union of the sets selected in both stages. The aim is to minimize the expected total cost of the solution.

The problem can be formulated as an integer program and the integrality constraints can be relaxed to yield the following linear program:

(SSC-P1):

$$\begin{aligned} \text{min:} \quad & \sum_{S \in \mathcal{S}} w_S^{(1)} x_S + \sum_A p_A \sum_{S \in \mathcal{S}} w_S^{(2)} r_{A,S} & \\ \text{subject to:} \quad & \sum_{S \in \mathcal{S}: e \in S} x_S + \sum_{S \in \mathcal{S}: e \in S} r_{A,S} \geq 1 & \text{for all } A, e \in A \\ & x_S, r_{A,S} \geq 0 & \text{for all } A, S \end{aligned}$$

Variables x_S indicate whether the set S is chosen in the first stage or not, and variables $r_{A,S}$ indicate if the set S is chosen in the second stage when scenario A realizes. The constraints only ensure that each element of A is covered in the first or second stage.

One can see that, when p_A is given explicitly for each scenario, one can solve exactly the above problem. However, the number of scenarios is exponential to the number of elements of the universe, and so an exact solution requires exponential time. Thus, we will try to approximate the solution of the above relaxation. Before that, however, we must make sure that seeking a solution for the relaxed program would help us find a good solution for the initial problem. The following theorem makes this explicit.

Theorem 4.1. *Suppose that we have a procedure that for every instance of (DSC) produces a solution of cost at most $p \cdot OPT_{Det}$, where OPT_{Det} is the cost of an optimal solution of the linear relaxation of (DSC). Then, one can convert any solution (x, r) of (SSC-P1) to an integer solution of cost at most $2p$ times the cost of (x, r). Thus, an optimal solution to (SSC-P1) gives a $2p$-approximation algorithm.*

Proof. Let $h(.)$ denote the objective function of (SSC-P1). We will argue that we can obtain an integer solution $(\tilde{x}, \tilde{r})$ of cost at most $2p \cdot h(x, r)$. Our rounding approach is simple. Observe that an element e is either covered to an extent of at least $1/2$ in the first stage by the variables x_S, or it is covered to an extent of at least $1/2$ by the variables $r_{A,S}$ in every scenario A containing e. Let $E = \{e : \sum_{S:e \in S} x_S \geq 1/2\}$, i.e. the elements that are covered by at least $1/2$ in the first stage. The vector $(2x)$ is then a fractional solution of the (DSC) instance with universe E, of cost $\sum_S 2w_S^{(1)} x_S$, and, thus, one can obtain a solution $\tilde{x}$ for this instance of cost at $p \cdot OPT_{Det}(E) \leq p \cdot \sum_S 2w_S^{(1)} x_S$. Similarly, for any scenario A, the vector $(2r_A)$ is a fractional solution of the (DSC) instance with universe $A \setminus E$, of cost $\sum_S 2w_S^{(2)} r_{A,S}$. Thus, one can obtain a solution $\tilde{r}_A$ for this instance of cost at most $p \cdot OPT_{Det}(A \setminus E) \leq p \cdot \sum_S 2w_S^{(2)} r_{A,S}$.

So, we have that $\sum_S w_S^{(1)} \cdot \tilde{x}_S \leq 2p \cdot \sum_S w_S^{(1)} x_S$ and $\sum_S w_S^{(2)} \tilde{r}_{A,S} \leq 2p \cdot \sum_S w_S^{(2)} r_{A,S} \Rightarrow p_A \sum_S w_S^{(2)} \tilde{r}_{A,S} \leq 2p \cdot p_A \sum_S w_S^{(2)} r_{A,S} \Rightarrow \sum_A p_A \sum_S w_S^{(2)} \tilde{r}_{A,S} \leq 2p \cdot \sum_A p_A \sum_S w_S^{(2)} r_{A,S}$. It is obvious now that if we output $\tilde{x}$ as the first-stage decisions vector, we get a solution of cost at most $2p \cdot h(x, r)$.

□

The above theorem also implies the following:

Corollary 4.1. *If the integrality gap of the linear relaxation of (DSC) is p, then the integrality gap of (SSC-P1) is at most $2p$.*

It is well-known ([8]) that the greedy algorithm for the deterministic set cover returns a solution of weight at most $\ln n \cdot OPT_{Det}$. Thus, the above theorem shows that if we could solve (SSC-P1), we would get a $2 \ln n$-approximation algorithm for the Stochastic Set Cover.

4.2.2 Towards a compact convex programming formulation

From what has already been mentioned, it seems difficult to find an optimal solution to (SSC-P1), since it has both an exponential number of variables and an exponential number of constraints; even writing out an optimal solution might take exponential space, and thus, time. So, we have to make another approach to our problem. Observe that the rounding process stated above involved only the examination of the first-stage vector x, and not the second-stage vectors r_A. This is important, as any rounding algorithm that needs information about each second-stage vector is essentially exponential, due to the exponential number of scenarios. In contrast, the above rounding process shows that if we could somehow solve the stochastic relaxation efficiently, then we would get a $2p$-approximation algorithm. We should remind the reader here that in any case, an efficient algorithm for a stochastic problem has to provide a first-stage decisions vector and a well-defined way to tackle any scenario that arrives. It is not necessary, and in fact it should be avoided, to precompute each decision that the algorithm will take for every scenario that may arrive. This observation motivates the following formulation of our problem:

(SSC-P2):

$$\text{min:} \quad \sum_{S \in \mathcal{S}} w_S^{(1)} x_S + f(x) \qquad \text{subject to } x_S \geq 0 \qquad \text{for all } S$$

$$\begin{aligned}
\text{where} \quad f(x) &= \sum_{A \subseteq U} p_A f_A(x), \\
\text{and} \quad f_A(x) &= \min \sum_{S \in \mathcal{S}} w_S^{(2)} r_{A,S} \\
&\quad \text{s.t:} \sum_{S \in \mathcal{S}: e \in S} r_{A,S} \geq 1 - \sum_{S \in \mathcal{S}: e \in S} x_S \qquad \text{for all } e \in A, \\
&\quad r_{A,S} \geq 0 \qquad \text{for all } S
\end{aligned}$$

It is straightforward to show that (SSC-P1) and (SSC-P2) are equivalent mathematical programs. The important thing however in the second formulation (SSC-P2) is that the objective value function is only a function of the first-stage variables and not the second-stage variables. More importantly, it expresses in the clearest way the whole idea behind 2-stage stochastic optimization: choose a first-stage vector, and when a specific scenario occurs, augment the first-stage solution to obtain a feasible solution for this scenario. Furthermore, each first-stage vector gives a different value to the objective function. This is why we seek the best possible one. The reader should however keep in mind that, in any case, we cannot even evaluate any objective function value, as this would require exponential time.

The polynomial number of variables of the above formulation, combined with the fact that the objective function of (SSC-P2) is convex, leads us to seek a convex optimization method to solve our problem. This is where the ellipsoid method fits in very well.

4.2.3 Solving the convex program

The ellipsoid method can generally be used to solve convex optimization programs (readers not familiar with the ellipsoid method are suggested to take a look at appendix B). In the ellipsoid method, we start by containing the feasible region within a ball and then generate a sequence of ellipsoids, each of successively smaller volume. In each iteration, one examines the center of the current ellipsoid and obtains a specific half-space defined by a hyperplane passing through the current ellipsoid center. If the current ellipsoid center is infeasible, then one uses a violated inequality as the hyperplane, otherwise, one uses an *objective function cut* to eliminate (some or all) feasible points whose objective function value is no better than the current center, and thus make progress. A new ellipsoid is then generated by finding the minimum-volume ellipsoid containing the half-ellipsoid obtained by the intersection of the current one with this half-space. Continuing this way, and using the fact that the volume of the successive ellipsoids decreases by a significant factor, one can show that after a polynomial number of iterations, the feasible point generated with the best objective function value is a near-optimal solution.

Let $\mathcal{P} = \mathcal{P}_0$ denote the initial polytope $\{x \in \mathbb{R}^m : 0 \leq x_S \leq 1 \text{ for all S}\}$, and x_i be the current iterate. If the current iterate x_i is feasible, then one could add the constraint $h(x) \leq h(x_i)$ while maintaining the convexity of the feasible region. But then, in subsequent iterations, one would need to check if the current iterate is feasible, and generate a separating hyperplane if not. However, without the ability to evaluate the objective function value, we cannot even decide whether the current point is feasible or not. Thus, finding a separating hyperplane seems to put a significant barrier in our attempt. An alternate approach is to use cuts generated by a subgradient, which can be seen as a measure of the "slope" of our functions and which essentially plays the role of the gradient when the function is not differentiable.

Note 4.1. *Before moving on, we should note that we will come across various algebraic notions from now on. The reader who is unfamiliar with them can take a look at appendix A.*

Definition 4.1. Let $g : \mathbb{R}^m \to \mathbb{R}$ be a function. We say that $d \in \mathbb{R}^m$ is a subgradient of g at the point u if $g(v) - g(u) \geq d \cdot (v - u), \forall v \in \mathbb{R}^m$.

If g is convex and differentiable, then its gradient at x is a subgradient. But a subgradient can exist even when g is not differentiable at x, as illustrated in figure 4.1. The same example shows that there can be more than one subgradients of a function g at a point x. A convex function has a subgradient at x if there is at least one non-vertical supporting hyperplane to **epi** g at $(x, g(x))$. This is the case, for example, if g is continuous. There are pathological convex functions which do not have subgradients at some points, but we will assume in the sequel that all convex functions that come up in our problems are subdifferentiable, which means that there exists at least one subgradient at every point x in **dom** g.

There are several ways to interpret a subgradient. A vector d is a subgradient of g at x if the affine function (of z) $g(x) + d \cdot (z - x)$ is a global underestimator of g. Geometrically, d is a subgradient of g at x if $(d, -1)$ supports **epi** g at $(x, g(x))$, as illustrated in figure 4.2.

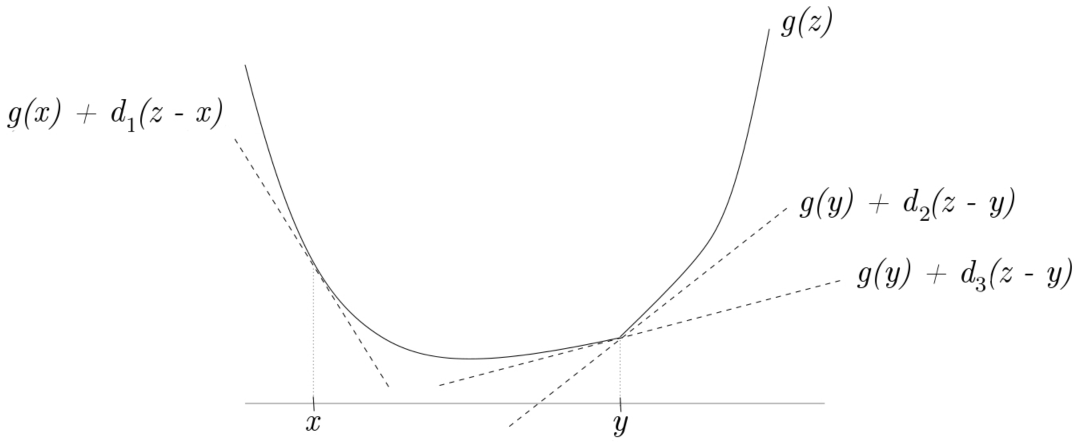

Figure 4.1: At x, the convex function g is differentiable, and d_1 (which is the derivative of g at x) is the unique subgradient at x. At the point y, g is not differentiable. At this point, g has many subgradients: two subgradients, d_2 and d_3, are shown.

Returning back to the ellipsoid method, one can see that if d_i is a subgradient at point x_i, then one could add the subgradient cut $d_i \cdot (x - x_i) \leq 0$ and proceed with the smaller polytope $\mathcal{P}_{i+1} = \mathcal{P}_i \cap \{x : d_i \cdot (x - x_i) \leq 0\}$. This is quite clear, because for any point x with $d_i \cdot (x - x_i) > 0$, we have that $g(x) \geq g(x_i) + d_i \cdot (x - x_i) > g(x_i)$, and thus such a point should be discarded, as we are dealing with a minimization problem.

Unfortunately, even computing a subgradient at a point x is hard to do in polynomial time for the objective functions that arise in stochastic programs. To deal with this problem, we are going to use approximate subgradients, as defined below:

Definition 4.2. We say that $\hat{d} \in \mathbb{R}^m$ is an $(\omega, \mathcal{D})$-subgradient of a function $g : \mathbb{R}^m \to \mathbb{R}$ at the point $u \in \mathcal{D}$ if $\forall v \in \mathcal{D}$, $g(v) - g(u) \geq \hat{d} \cdot (v - u) - \omega g(u)$.

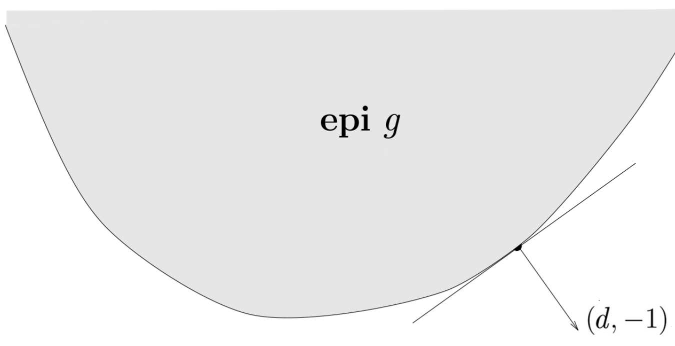

Figure 4.2: A vector $d \in \mathbb{R}^n$ is a subgradient of g at x if and only if $(d, -1)$ defines a supporting hyperplane to **epi** g at $(x, g(x))$.

We will only use $(\omega, \mathcal{P})$-subgradients in the algorithm, which we abbreviate and denote as ω-subgradients from now on. We will show that one can compute, with high probability, an ω-subgradient of $h(.)$ at any point x, by sampling from the black-box on scenarios. At a feasible point x_i, we compute an ω-subgradient $\hat{d}_i$ and add the inequality $\hat{d}_i \cdot (x - x_i) \leq 0$ to chop off a region of $\mathcal{P}_i$ and get the polytope $\mathcal{P}_{i+1}$. The first problem that rises is that by using an approximate subgradient cut, we might discard points with objective function value better than at the current iterate x_i. However, we overcome this by showing that at a discarded point the function value is not much better than $h(x_i)$. Continuing this way, we obtain a polynomial number of points $x_0, x_1, ..., x_k$ such that $x_i \in \mathcal{P}_i \subseteq \mathcal{P}_{i-1}$ for each i, and the volume of $\mathcal{P}_k$ is "small". Now, if the function $h(.)$ has bounded variation on nearby points, then we can show that $\min_i h(x_i)$ is close to the optimal value $h(x^*)$ with high probability.

One last hurdle remains: how to compute $\min_i h(x_i)$, and thus $\bar{x} = \arg\min_i h(x_i)$, when we cannot even compute h at any point. Nonetheless, we manage to overcome this, too, by computing a point $\bar{x}$ in the convex hull of $x_0, x_1, ..., x_k$, at which the objective function value is close to $\min_i h(x_i)$. We are going to use approximate subgradients here, too. At the heart of this procedure is a subroutine that given two points y_1, y_2, returns a point y on the line segment connecting y_1 and y_2 such that $h(y)$ is close to $\min(h(y_1), h(y_2))$. Such a y is obtained by performing a bisection search, using the subgradient to infer which direction to move along the line segment. Iterating on this subroutine, we finally get a point $\bar{x}$ such that $h(\bar{x})$ is close to $\min_i h(x_i)$.

We are now ready to proceed in the full analysis of our algorithm. Although it is quite extensive, we believe it is worth studying as many interesting ideas are introduced in the attempt to overcome the various obstacles on the way.

4.2.4 Algorithm details and analysis

We are going to describe the algorithm for an arbitrary convex function $h(.)$ and an arbitrary (rational) polytope $\mathcal{P}$. Let $OPT = \min\{h(x) : x \in \mathcal{P}\}$ denote the optimal solution value. We use $\|u\|$ to denote the l_2 norm of u, i.e. $(\sum u_i^2)^{1/2}$.

Throughout our analysis, we are several times going to use the convention that the objective function has Lipschitz constant. It is sensible for the problems we are modeling, and we will actually prove that this is the case for the functions we are studying.

Definition 4.3. Given a function $g : \mathbb{R}^m \to \mathbb{R}$, we say that g has Lipschitz constant (at most) K, if $|g(v) - g(u)| \leq K\|v - u\|$ for all $u, v \in \mathbb{R}^m$.

Lipschitz constant makes precise the notion of bounded variation, which is crucial in our analysis. Note that a function that has a Lipschitz constant is continuous, but is not necessarily differentiable.

Let our objective function $h : \mathbb{R}^m \to \mathbb{R}$ have Lipschitz constant K. We assume that $x \geq \mathbf{0}$ is a valid inequality for $\mathcal{P}$. We also assume that the polytope $\mathcal{P}$ is contained in the ball $B(\mathbf{0}, R) = \{x : \|x\| \leq R\}$, and contains a ball of radius r such that $\ln R$ and $\ln(1/r)$ are polynomially bounded. These last two conditions are needed for the ellipsoid algorithm. For all the optimization problems we consider, our initial polytope is $\mathcal{P} = \{x \in \mathbb{R}^m : 0 \leq x_S \leq 1\}$ and, so, it is trivial to find such R and r. More details about these can be found in [14]. Finally, for reasons that will be made clear in the following, we set $V = \min(1, r)$ and define $\lambda = \max\left(1, \max_S \frac{w_S^{(2)}}{w_S^{(1)}}\right)$. We assume that λ or an upper bound on it is known to the algorithm.

The basic procedure we will use is `FindOpt`, which takes two parameters γ and ε and returns a feasible solution $\bar{x}$ such that $h(\bar{x}) \leq OPT/(1-\gamma) + \varepsilon$, in time polynomial in the dimension m and $\ln(\frac{KRm}{V\varepsilon})$ (excluding the time to compute the ω-subgradients, which we will discuss later on). This is the main procedure that uses the ellipsoid method and the notion of ω-subgradients to get close to an optimal solution. It also uses a subroutine `FindMin` which takes a set of feasible points $x_0, x_1, ..., x_k$, and returns a feasible point having function value close to $\min_i h(x_i)$. The main problem here, as we have already mentioned, is that we cannot evaluate h at any point x. So we are seeking a point $\bar{x}$ in the convex hull of $x_0, ..., x_k$, such that we can prove that $h(\bar{x})$ is close to $\min_{i=0}^k h(x_i)$.

We give now a formal description of procedures `FindOpt` and `FindMin`, together with pictorial descriptions in figures 4.3, and 4.4 and 4.5, respectively.

Procedure `FindOpt`(γ, ε)

```
*(Returns a point $\bar{x}$ such that $h(\bar{x}) \leq OPT/(1-\gamma) + \varepsilon$. Assume $\gamma \leq 1/2$.)*
**Set** $k \leftarrow 0$,  $y_0 \leftarrow \mathbf{0}$,  $N \leftarrow \lceil 2m^2 \ln(\frac{16KR^2}{V\varepsilon}) \rceil$,  $n \leftarrow N \log_2(\frac{8NKR}{\varepsilon})$,  $\omega \leftarrow \gamma/(2n)$.
Let $E_0 \leftarrow B(\mathbf{0}, R)$ and $\mathcal{P}_0 \leftarrow \mathcal{P}$.

**for** $i = 0, ..., N$ **do**
    *[We maintain the invariant that $E_i$ is an ellipsoid centered at $y_i$ containing the current polytope $P_k$.]*
    **if** $y_i \in P_k$ **then**
        **Set** $x_k \leftarrow y_i$. Let $\hat{d}_k$ be an $\omega$-subgradient of $h(.)$ at $x_k$. Let $H$ denote the half space $\{x \in \mathbb{R}^m : \hat{d}_k \cdot (x - x_k) \leq 0\}$. **Set** $\mathcal{P}_{k+1} \leftarrow \mathcal{P}_k \cap H$ and $k \leftarrow k+1$.
    **else**
        Let $a \cdot x \leq b$ be a violated inequality, that is, $a \cdot y_i > b$, whereas $a \cdot x \leq b$ for all $x \in \mathcal{P}_k$. Let $H$ be the half-space $\{x \in \mathbb{R}^m : a \cdot (x - y_i) \leq 0\}$.
    **Set** $E_{i+1}$ to be the ellipsoid of minimum volume containing the half-ellipsoid $E_i \cap H$.

**Set** $k \leftarrow k - 1$. We now have a collection of points $x_0, ..., x_k$ such that each $x_l \in \mathcal{P}_l \subseteq \mathcal{P}_{l-1}$.
**Return** FindMin$(\omega; x_0, ..., x_k)$.
```

Before analyzing the above procedures, we mention some well known facts that we are going to use. (see [14])

Lemma 4.1. *The volume of the ball $B(u, D) = \{x \in \mathbb{R}^m : \|x - u\| \leq D\}$ where $u \in \mathbb{R}^m, D \geq 0$, is D^m vol$(B(\mathbf{0}, 1))$.*

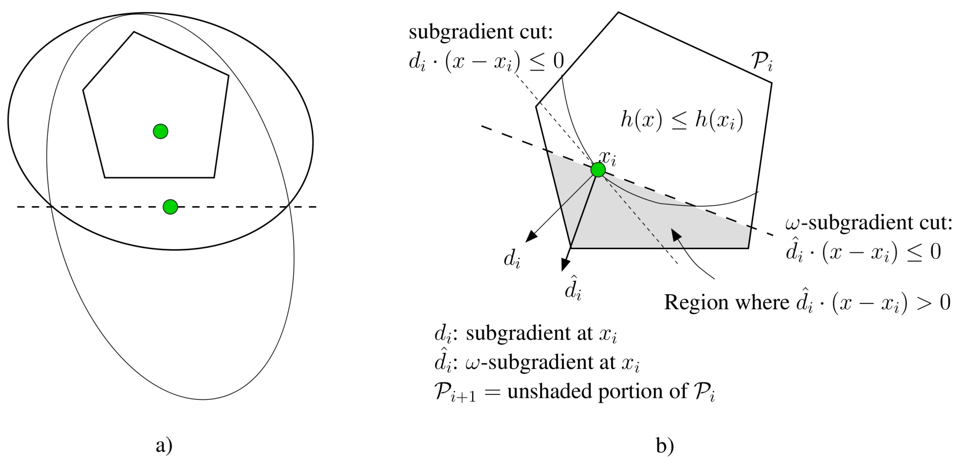
subgradient cut:
$d_i \cdot (x - x_i) \leq 0$
$\mathcal{P}_i$
$h(x) \leq h(x_i)$
x_i
ω-subgradient cut:
$\hat{d}_i \cdot (x - x_i) \leq 0$
d_i
$\hat{d}_i$
Region where $\hat{d}_i \cdot (x - x_i) > 0$
d_i: subgradient at x_i
$\hat{d}_i$: ω-subgradient at x_i
$\mathcal{P}_{i+1}$ = unshaded portion of $\mathcal{P}_i$
a)
b)

Figure 4.3: a) Cut derived from a violated inequality. b) Subgradient and ω-subgradient cuts.

Procedure `FindMin`$(\omega; x_0, ..., x_k)$

```
Set $p \leftarrow \varepsilon/(4k)$,   $\bar{x} \leftarrow x_0$,   $N' \leftarrow \lceil \log_2(\frac{8kKR}{\varepsilon}) \rceil$.
for $i = 1, ..., k$ do
    [We maintain the invariant that $h(\bar{x}) \leq (\min_{l=0}^{i-1} h(x_l) + (i-1)p)/(1-\omega)^{(i-1)N'}$.]
    Set $y_1 \leftarrow \bar{x}$,   $y_2 \leftarrow x_i$.
    [We use binary search to find $y$ on the $\bar{x} - x_i$ line segment with value close to $\min(h(\bar{x}), h(x_i))$.]
    for $j = 1, ..., N'$ do
        [We maintain that $h(y_1) \leq h(\bar{x})/(1-\omega)^{j-1}$, $h(y_2) \leq h(x_i)/(1-\omega)^{j-1}$.]
        Set $y \leftarrow \frac{y_1+y_2}{2}$. Compute an $\omega$-subgradient $\hat{d}$ of $h$ at the point $y$.
        if $\hat{d} \cdot (y_1 - y_2) = 0$ then
            exit current loop
        if $\hat{d} \cdot (y_1 - y_2) > 0$ then
            set $y_1 \leftarrow y$
        else
            set $y_2 \leftarrow y$
    Set $\bar{x} \leftarrow y$.
Return $\bar{x}$.
```

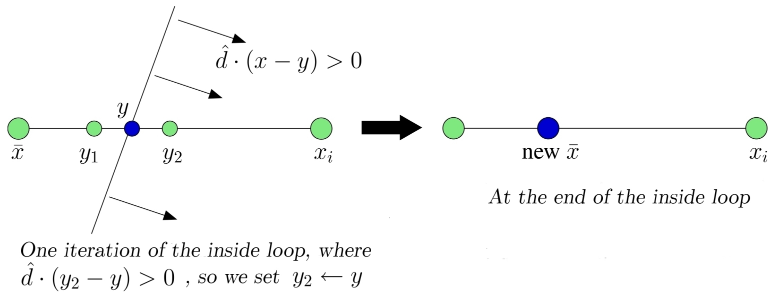
$\hat{d} \cdot (x - y) > 0$
y
$\bar{x}$
y_1
y_2
x_i
new $\bar{x}$
x_i
At the end of the inside loop
One iteration of the inside loop, where
$\hat{d} \cdot (y_2 - y) > 0$, so we set $y_2 \leftarrow y$

Figure 4.4: Inside loop of procedure `FindMin`.

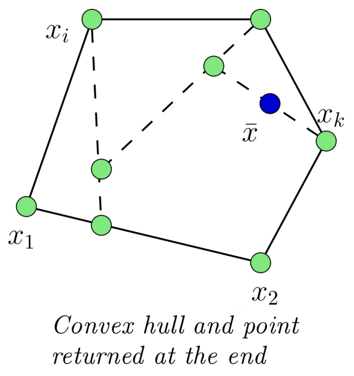

Figure 4.5: Point returned at the end of the outside loop of procedure `FindMin`.

Lemma 4.2. *Let $E \subseteq \mathbb{R}^m$ be an ellipsoid and $H \subseteq \mathbb{R}^m$ be a half-space passing through the center of E. Then there is a unique ellipsoid E' of minimum volume containing the half-ellipsoid $E \cap H$ and $\frac{vol(E')}{vol(E)} \leq e^{-1/(2m)}$.*

Lemma 4.3. *Let $T : \mathbb{R}^m \to \mathbb{R}^m$ be an affine transformation with $T(x) = Qx + t$, where $det\, Q \neq 0$. Then for any set $S \subseteq \mathbb{R}^m$ we have $vol(T(S)) = |det\, Q| vol(S)$.*

We are now going to prove some lemmas that will lead to our main result. We should observe that the invariant of the for-loop of procedure `FindOpt` is clearly maintained.

Lemma 4.4. *The points $x_0, ..., x_k$ generated by procedure* `FindOpt` *satisfy $\min_{i=0}^{k} h(x_i) \leq (OPT + \frac{\varepsilon}{4})/(1-\omega)$.*

Proof. Let x^* be an optimal solution, that is, $h(x^*) = OPT$. If $\hat{d}_l \cdot (x^* - x_l) \geq 0$ for some l, then x^* was discarded with the ω-subgradient cut we used in that step. We have that $h(x^*) - h(x_l) \geq \hat{d}_l \cdot (x^* - x_l) - \omega h(x_l)$, and thus $h(x^*) - h(x_l) + \omega h(x_l) \geq 0$, which means that $h(x_l) \leq h(x^*)/(1-\omega) < (h(x^*) + \frac{\varepsilon}{4})/(1-\omega)$. Otherwise, we have that x^* belongs to every polytope $\mathcal{P}_i$ generated by the procedure. We are going to use a scaling argument here. We will "shrink" the initial polytope around x^* and prove that function h has value near to the optimal in every point of this shrunken polytope. We will then prove that the last polytope $\mathcal{P}_k$ produced by our procedure has a boundary generated by a hyperplane $\hat{d}_l \cdot (x - x_l)$ that belongs to the shrunken polytope, and so we can bound the value of function h at point x_l using this fact. We are now going to formalize all these ideas. A visualization of these is shown in figure 4.6.

Let $t = \frac{\varepsilon}{8KR}$, and consider the affine transformation $T(x) = tI_m(x - x^*) + x^* = tx + (1-t)x^*$, where I_m is the $m \times m$ identity matrix. Let $W = T(\mathcal{P})$, so W is a shrunken version of $\mathcal{P}$ "centered" around x^*. Observe the following facts:

1. $W \subseteq \mathcal{P}$, because $\mathcal{P}$ is convex, and any point $T(x) \in W$ is a convex combination of x and x^*, which both belong to $\mathcal{P}$, and so $T(x) \in \mathcal{P}$.

2. $vol(W) = vol(T(\mathcal{P})) = |\det(tI_m)| vol(\mathcal{P}) = t^m\, vol(\mathcal{P}) \geq (tV)^m\, vol(B(\mathbf{0}, 1))$, using lemmas 4.1 and 4.3 and the fact that $\mathcal{P}$ contains a ball of radius V by assumption.

3. For any $y = T(x) \in W$, $\|y - x^*\| = t\|x - x^*\| \leq \frac{\varepsilon}{8KR} \cdot (2R) \leq \frac{\varepsilon}{4K}$, since $x, x^* \in \mathcal{P}$ and $\mathcal{P} \subseteq B(\mathbf{0}, R)$. And since h has Lipschitz constant K, we get that $|h(y) - h(x^*)| \leq K\|y - x^*\| \leq \frac{\varepsilon}{4}$. Using now the fact that $h(x^*)$ is the minimum value of h, we get that $h(y) \geq h(x^*)$, and, thus, $h(y) \leq h(x^*) + \frac{\varepsilon}{4}$.

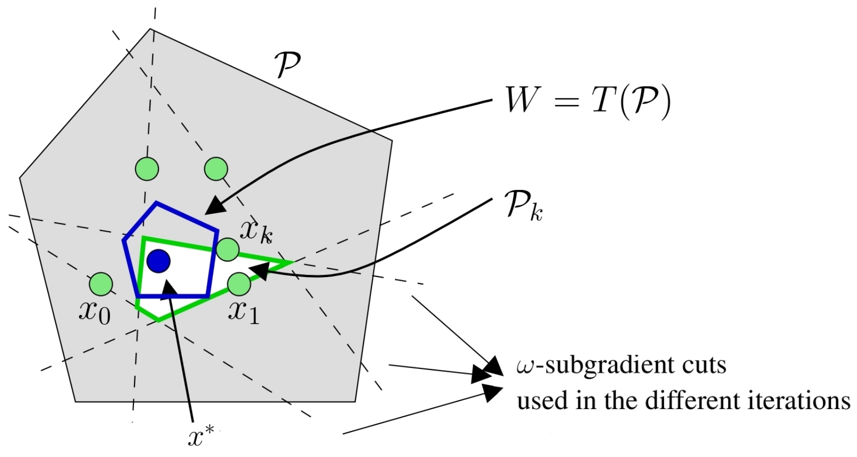

Figure 4.6: Pictorial description of the scaling argument used in the proof of lemma 4.4.

We are now going to bound the volume of $\mathcal{P}_k$. Since $\frac{vol(E_{i+1})}{vol(E_i)} \leq e^{-1/(2m)}$ for every i (lemma 4.2), and the volume of the initial ball $E_0 = B(\mathbf{0}, R)$ that contains $\mathcal{P}$ is $R^m vol(B(\mathbf{O}, 1))$, we get that

$$\begin{aligned}
vol(\mathcal{P}_k) &\leq vol(E_N) \\
&\leq e^{-N/(2m)} vol(E_0) \\
&= \left(\frac{1}{e}\right)^{\lceil 2m^2 \ln(\frac{16KR^2}{V\varepsilon}) \rceil/(2m)} R^m vol(B(\mathbf{O},1)) \\
&\leq \left(\frac{1}{e}\right)^{2m^2 \ln(\frac{16KR^2}{V\varepsilon})/(2m)} R^m vol(B(\mathbf{O},1)) \\
&= \left(\frac{V\varepsilon}{16KR^2}\right)^m R^m vol(B(\mathbf{O},1)) \\
&= \left(\frac{V\varepsilon}{16KR}\right)^m vol(B(\mathbf{O},1))
\end{aligned}$$

and using the fact that $t = \frac{\varepsilon}{8KR}$ we get that

$$vol(\mathcal{P}_k) \leq \left(\frac{Vt}{2}\right)^m vol(B(\mathbf{O},1)) < (tV)^m \, vol(B(\mathbf{0},1)) \leq vol(W).$$

Thus, $vol(\mathcal{P}_k) < vol(W)$, and since x^* belongs in both sets, there must be a point $y \in W$ that lies on a boundary of $\mathcal{P}_k$ generated by a hyperplane $\hat{d}_l \cdot (x - x_l) = 0$, otherwise W would be a subset of $\mathcal{P}_k$, which is impossible due to the volume inequality. This implies that $\hat{d}_l \cdot (y - x_l) = 0$, and since $h(y) - h(x_l) \geq \hat{d}_l \cdot (y - x_l) - \omega h(x_l)$, we get that $h(x_l) \leq h(y)/(1-\omega)$. Using now the fact that for any $z \in W$, $h(z) \leq h(x^*) + \frac{\varepsilon}{4}$ (proved above), and since $y \in W$, we get that $h(x_l) \leq (h(x^*) + \frac{\varepsilon}{4})/(1-\omega)$.

□

Lemma 4.5. *Procedure* `FindMin` *returns a point* $\bar{x}$ *such that* $h(\bar{x}) \leq \left[(\min_{i=0}^{k} h(x_i)) + \frac{\varepsilon}{4}\right] / (1-\omega)^{kN'}$.

We are only going to give an outline of the proof, and not the formal proof itself, which can be found in [38]. The main problem is that we cannot evaluate h at any point x. To overcome this, we are seeking a point $\bar{x}$ in the convex hull of $x_0, ..., x_k$, such that we can prove that $h(\bar{x})$ is close to $\min_{i=0}^{k} h(x_i)$. More precisely, we perform a bisection search using ω-subgradients on the line segment connecting points $\bar{x}$ and x_i, where $\bar{x}$ is close to $\min_{l=0}^{i-1} h(x_l)$, and we iterate for all points x_i. The final $\bar{x}$ returned satisfies the above property. To prove this, we use induction on the invariant conditions given as comments in the description of the procedure.

We are now going to prove the first significant result that will lead to our FPRAS.

Theorem 4.2. *Procedure* `FindOpt` *returns a feasible point* $\bar{x}$ *which satisfies* $h(\bar{x}) \leq OPT\ / (1-\gamma)+\varepsilon$, *in time* $O\left(T(\omega)\cdot m^2 \ln^2(\frac{KRm}{V\varepsilon})\right)$, *where* $T(\omega)$ *denotes the time taken to compute an* ω*-subgradient. Here* ω *has to be sufficiently small, more precisely* $\omega = \Theta\left(\gamma/(m^2 \ln^2(\frac{KRm}{V\varepsilon}))\right)$.

Proof. By Lemmas 4.4 and 4.5, we get that

$$\begin{aligned} h(\bar{x}) &\leq \left(\frac{OPT+\varepsilon/4}{1-\omega}+\frac{\varepsilon}{4}\right)/(1-\omega)^{kN'} \\ &= \left(OPT+\frac{\varepsilon}{4}+(1-\omega)\frac{\varepsilon}{4}\right)/(1-\omega)^{kN'+1} \\ &\leq \left(OPT+\frac{\varepsilon}{2}\right)/(1-\omega)^{kN'+1} \end{aligned}$$

Since $kN' \leq N\lceil \log_2(\frac{8NKR}{\varepsilon})\rceil$, we get that $kN' \leq N\log_2(\frac{8NKR}{\varepsilon})+N$. Remember that $N\log_2(\frac{8NKR}{\varepsilon}) = n$. Thus, we get that $(1-\omega)^{kN'+1} \geq (1-\omega)^{n+N+1}$.

We will use the inequality $(1-x)^n \geq 1-nx$, $\forall x \in (0,1)$, that is true even if $n \geq 1$ is not integer. We can easily prove this by studying the function (of n) $(1-x)^n+nx-1$. So, we now have that

$$\begin{aligned} (1-\omega)^{kN'+1} &\geq (1-\omega)^{n+N+1} \\ &\geq 1-(n+N+1)\omega \\ &= 1-\frac{n+N+1}{2n}\cdot\gamma \\ &\geq (1-\gamma) \\ &\geq \frac{1}{2} \qquad \text{(since we assumed that } \gamma \leq \frac{1}{2}) \end{aligned}$$

We have used the fact that $n \geq N+1$ (observe that $n-N-1 = N\log_2(\frac{8NKR}{\varepsilon})-N-1 = N\log_2(\frac{8NKR}{2\varepsilon})-1 \geq 0$), and, thus, $(n+N+1)/(2n) = 1/2+(N+1)/2n \leq 1/2+1/2 \leq 1$. Thus, we get that

$$\begin{aligned} h(\bar{x}) &\leq \left(OPT+\frac{\varepsilon}{2}\right)/(1-\gamma) \\ &= OPT/(1-\gamma)+\frac{\varepsilon}{2(1-\gamma)} \\ &\leq OPT/(1-\gamma)+\varepsilon. \end{aligned}$$

As for the running time, in each iteration of the main loop of `FindOpt` we do at most one calculation of an ω-subgradient. Thus, the time cost of the loop is $O(N\cdot T(\omega))$. The procedure `FindMin` takes time $O(kN'\cdot T(\omega))$. And, as proved above, $kN' \leq n+N$, and, thus, the total time required is $O((n+N)\cdot T(\omega))$. We have that $n = N\log_2(\frac{8NKR}{\varepsilon})$ and $N \leftarrow \lceil 2m^2\ln(\frac{16KR^2}{V\varepsilon})\rceil$, and so we get that $n = \Theta\left(2m^2\cdot\ln^2(\frac{KRm}{V\varepsilon})\right)$. This also determines the values of $\omega = \gamma/(2n)$ and gives $\omega = \Theta\left(\gamma/(m^2\cdot\ln^2(\frac{KRm}{V\varepsilon}))\right)$. Plugging things together, we obtain a running time $O((n+N)\cdot T(\omega)) = O\left(m^2\ln^2(\frac{KRm}{V\varepsilon})\cdot T(\omega)\right)$.

□

We are now going to convert the above guarantee into a purely multiplicative $(1 + k)$-guarantee. To do this, we need to avoid situations where $OPT = 0$, and, so, we need to obtain a lower bound on OPT. At this point, we make the mild assumption that the cost of every set S, in stage I and in every stage II scenario, is at least 1. For integer costs, this simply means that the cost of any non-null scenario is not 0. This helps us detect, by sampling $O(\lambda)$ times, whether the probability that some scenario $A \neq \emptyset$ occurs is at least $1/\lambda$. If so, then $OPT \geq 1/\lambda$, otherwise $x = \mathbf{0}$ is an optimal solution (with high probability). More specifically, we use the procedure `ConvOpt`, which samples and determines with high probability that either $x = \mathbf{0}$ is an optimal solution, and returns this solution, or obtains a lower bound on OPT and then calls `FindOpt` setting γ and ε appropriately.

```
Procedure ConvOpt($k, \delta$)
    (Returns a point $\bar{x}$ such that $h(\bar{x}) \leq (1+k) \cdot OPT$ with probability at least $1-\delta$. Assume $\delta \leq 1/2$.)
    Define $\lambda = \max(1, \max_S w_S^{(2)}/w_S^{(1)})$.
    if $\ln(1/\delta) \in \mathbb{N}$ then
        set $\delta' \leftarrow \delta$
    else
        set $\delta' \leftarrow e^{\frac{-\lceil \lambda \ln(1/\delta) \rceil}{\lambda}}$
    Sample $M = \lambda \ln(\frac{1}{\delta'}) = \lceil \lambda \ln(\frac{1}{\delta}) \rceil$ times from the distribution on scenarios.
    Let $X$ = number of times a non-null scenario occurs.

    if $X = 0$ then
        return $x = \mathbf{0}$
    else
        [With high probability, $OPT \geq p/\lambda$, where $p = \frac{\delta'}{\ln(1/\delta')}$ ]
        set $\varepsilon \leftarrow kp/(2\lambda)$,    $\gamma \leftarrow k/3$
        return FindOpt($\gamma, \varepsilon$)
```

Lemma 4.6. *Procedure* `ConvOpt` *determines (correctly) with probability at least $1 - \delta$, that $OPT \geq p/\lambda$, or that $x = \mathbf{0}$ is an optimal solution.*

Proof. Observe that we do not always use the input value δ, but a smaller one, δ'. We make this slight change in the input data so that the number of samples given by the algorithm is always a natural number (it also makes some technical details of the proof easier to handle). In any case, we achieve a guarantee of at least $1 - \delta'$, and since $\delta' \leq \delta \Rightarrow 1 - \delta' \geq 1 - \delta$, we get the desired guarantee of at least $1 - \delta$. Also, note that $p \leq 1$ since $\delta' \leq \delta \leq 1/2$. Since we incur a cost of at least 1 in every non-null scenario, and the probability of occurrence of a non-null scenario is $q = \sum_{A \subseteq U, A \neq \emptyset} p_A$, we get that $OPT \geq q$. Let $r = \Pr[X = 0] = (1 - q)^M$. Since $\forall x,\ e^x \geq 1 + x$, we get that $e^{-x} \geq 1 - x$ and, thus, $e^{-q} \geq 1 - q \geq 0 \Rightarrow e^{-qM} \geq (1 - q)^M$. So, we have that $r \leq e^{-qM}$. We also get that $r \geq 1 - qM$, by using again the inequality $(1 - x)^n \geq 1 - nx,\ x \in (0, 1),\ n \in \mathbb{R}_{\geq 1}$.

We now check three cases:

- If $q \geq 1/\lambda$, then $r \leq e^{-(1/\lambda)M} = e^{-(1/\lambda)\lambda \ln(\frac{1}{\delta'})} = e^{-\ln(\frac{1}{\delta'})} = \delta' \leq \delta$. Now, since $OPT \geq q$, we get that $OPT \geq 1/\lambda$, and, since $p \leq 1$, we can say that with probability at least $1 - \delta$, we have that $OPT \geq p/\lambda$.
- If $q \leq \delta'/M \quad (= \delta'/(\lambda \ln(\frac{1}{\delta'})) = p/\lambda \leq 1/\lambda)$, then $r \geq 1 - \delta' \geq 1 - \delta$. We now return $x = \mathbf{0}$ as an optimal solution with probability at least $1 - \delta$, which is an optimal solution,

because $q \leq 1/\lambda$ implies that it is always at least as good to defer to stage II, since the expected stage II cost of a set S is at most $q \cdot w_S^{(2)} \leq w_S^{(1)}$. To make this more clear, observe that if we decide not to choose a set S in stage I, then the expected cost incurred in stage II for this decision is $(\sum p_A r_{A,S}) \cdot w_S^{(2)}$. Even if $r_{A,S} = 1\ \forall A$, we get that $(\sum p_A r_{A,S}) \cdot w_S^{(2)} \leq (\sum p_A) \cdot w_S^{(2)} \leq q \cdot w_S^{(2)} \leq w_S^{(1)}$, and, thus, there is no point in choosing this set in stage I. So, $x = \mathbf{0}$ is indeed an optimal solution.

- If $\delta'/M < q < 1/\lambda$, then we always return a correct answer, since it is both true that $x = \mathbf{0}$ is an optimal solution (because we still have that $q < 1/\lambda$), and $OPT \geq q > \delta'/M = \delta'/(\lambda \ln(\frac{1}{\delta'})) = p/\lambda$.

□

We are now going to focus on the computation of ω-subgradients, and we will at the end prove that the algorithm `ConvOpt` returns a $(1+k)$-optimal solution with probability at least $1-\delta$ in polynomial time.

Recall that our objective function is $h(x) = w^{(1)} \cdot x + f(x)$, where $f(x) = \sum_{A \subseteq U} p_A f_A(x)$, and

$$\begin{array}{rcll} f_A(x) & = & \min \sum_{S \in \mathbf{S}} w_S^{(2)} r_{A,S} & \\ & & \text{s.t:} \sum_{S \in \mathbf{S}: e \in S} r_{A,S} \geq 1 - \sum_{S \in \mathbf{S}: e \in S} x_S & \text{for all } e \in A, \\ & & r_{A,S} \geq 0 & \text{for all } S \end{array}$$

We are now going to present the dual of the above program, which we will use during our analysis. For an introduction to linear programming and duality, the reader can refer to [29]. By taking the dual we can write

$$\begin{array}{rcll} f_A(x) & = & \max \sum_{e \in U} \big(1 - \sum_{S: e \in S} x_S\big) z_{A,e} & \\ & & \text{s.t:} \sum_{e \in S} z_{A,e} \leq w_S^{(2)} & \text{for all } S \in \mathcal{S}, \\ & & z_{A,e} = 0 & \text{for all } e \notin A, \\ & & z_{A,e} \geq 0 & \text{for all } e \end{array}$$

We now remind the reader about the two assumptions that we have taken and have already mentioned: (i) $\lambda = \max\big(1, \max_S w_S^{(2)}/w_S^{(1)}\big)$ is known, and (ii) function h has Lipschitz constant K. The latter will indeed be proved to be valid for the functions that we study.

Our approach to computing ω-subgradients is the following:

- At first, we prove that any vector $\hat{d}$ that component-wise approximates a subgradient at x to within a certain accuracy is an ω-subgradient at x.
- Next, we show that at every point x there exists a "nice" subgradient with components $d_S \in [-w_S^{(2)}, w_S^{(1)}]$. This gives a bound on the Lipschitz constant and allows us to compute an ω-subgradient by using a sampling procedure.
- Finally, we show that we can approximate this "nice" subgradient by using the aforementioned sampling with high probability.

We will then be in a position to prove that the algorithm `ConvOpt` gives an FPRAS for our problem.

Lemma 4.7. *Let d be a subgradient of $h(.)$ at the point $x \in \mathcal{P}$, and suppose that $\hat{d}$ is a vector such that $d_S - \omega w_S^{(1)} \leq \hat{d}_S \leq d_S$ for all S. Then $\hat{d}$ is an ω-subgradient of $h(.)$ at x.*

The proof is pretty straightforward and can be found in appendix C.

Lemma 4.8. *Consider any point $x \in \mathcal{P}$ and let z_A^* be an optimal dual solution for scenario A with x as the stage I vector. Then the vector d with components $d_S = w_S^{(1)} - \sum_A p_A \sum_{e \in S} z_{A,e}^*$ is a subgradient at x, and $\|d\| \leq \lambda \|w^{(1)}\|$.*

The proof can be found in appendix C.

Lemma 4.9. *Suppose $\|d(x)\| \leq D$ for every x, where $d(x)$ is a subgradient of $h(.)$ at point x. Then $h(.)$ has Lipschitz constant (at most) D.*

Proof. Consider any two points $u, v \in \mathbb{R}^m$. Let d, d' denote the subgradients at u, v respectively, with $\|d\|, \|d'\| \leq D$. Then we have $h(v) - h(u) \geq d \cdot (v - u) \geq -\|d\| \, \|v - u\| \geq -D\|v - u\|$, and similarly $h(u) - h(v) \geq -\|d'\| \, \|u - v\| \geq -D\|u - v\| \Rightarrow h(v) - h(u) \leq D\|v - u\|$. So, we get that $|h(v) - h(u)| \leq D\|v - u\|$.

□

Lemmas 4.8 and 4.9 show that the functions that we study have Lipschitz constants that can be bound by $\lambda \|w^{(1)}\|$, which is a known real number for our problem. Note also that the Lipschitz constant K of our functions is such that $\ln K$ is polynomially bounded.

We will now prove the following sampling lemma which shows that we can efficiently compute an ω-subgradient of $h(.)$ at any point x.

Lemma 4.10. *Let $X \in [-a, b]$ be a random variable, $a, b > 0$, computed by sampling from a probability distribution π. Let $\mu = \mathbf{E}[X]$ and $k = \max(1, a/b)$. Then for any $c > 0$, by taking $\frac{100k^2}{3c^2} \ln\left(\frac{1}{\delta}\right)$ independent samples from π, one can compute an estimate Y such that $\mu - 2c \cdot b \leq Y \leq \mu$ with probability at least $1 - \delta$.*

Proof. Let $q = \max(a, b)$. The variance of X is $\sigma^2 = \mathbf{E}[X^2] - \mu^2 \leq q^2$, since $X^2 \in [0, q^2] \Rightarrow \mathbf{E}[X^2] \leq q^2$. We divide the samples we have taken into $s_1 = \frac{20}{3} \ln\left(\frac{1}{\delta}\right)$ groups, each containing $s_2 = 5k^2/c^2$ samples. Let X_{ij} be the value of X computed from the j^{th} sample of group i. Let Y_i be the average of the X_{ij} values of group i, i.e., $Y_i = (\sum_{j=1}^{s_2} X_{ij})/s_2$. We set $Y = \text{median}(Y_1, ..., Y_{s_1}) - c \cdot b$. Observe that the variables X_{ij} are iid with mean μ and variance σ^2. So we have that $\mathbf{E}[Y_i] = (\sum_{j=1}^{s_2} \mathbf{E}[X_{ij}])/s_2 = \mu$ and $\text{Var}[Y_i] = \text{Var}\left[\frac{X_{i1}}{s_2} + ... + \frac{X_{is_2}}{s_2}\right] = s_2 \cdot \frac{\sigma^2}{s_2^2} = \frac{\sigma^2}{s_2}$. By Chebyshev's inequality, we get $\Pr[|Y_i - \mu| > c \cdot b] \leq \frac{\sigma^2}{s_2 (cb)^2} \leq \frac{q^2}{s_2 c^2 b^2}$. If $a > b$ then $q = a$ and $k = a/b$, else $q = b$ and $k = 1$. In any case, we get $\Pr[|Y_i - \mu| > c \cdot b] \leq \frac{k^2}{s_2 c^2} = \frac{1}{5}$.

Now let $Z_i = 1$ if $|Y_i - \mu| > c \cdot b$, and 0 otherwise, and $Z = \sum_{i=1}^{s_1} Z_i$. Then $\mathbf{E}[Z] = \sum_{i=1}^{s_1} \mathbf{E}[Z_i] = s_1 \cdot \Pr[|Y_i - \mu| > c \cdot b] \leq \frac{s_1}{5}$ and the variables Z_i are independent. If $Y > \mu$ or $Y < \mu - 2c \cdot b$, then at least $s_1/2$ variables Z_i must be set to 1. Therefore, by Chernoff bounds we have $\Pr\left[Y \notin [\mu - 2c \cdot b, \mu]\right] \leq e^{-\frac{3s_1}{20}} \leq \delta$.

□

Corollary 4.2. *At any point $x \in \mathcal{P}$, one can compute an ω-subgradient with probability at least $1 - \delta$ using at most $T(\omega) = \frac{400\lambda^2}{3\omega^2} \ln\left(\frac{m}{\delta}\right)$ independent samples from the probability distribution on scenarios.*

Proof. The proof is an easy corollary of lemmas 4.8, 4.9 and 4.10. We use the sampling process described in lemma 4.10. Each time we sample and get a scenario A, we compute the quantities $X_S = w_S^{(1)} - \sum_{e \in S} z_{A,e}^*$, where z_A^* is an optimal dual solution (which can be computed in polynomial time) for scenario A with x as the first-stage vector. Observe that the vector d with components $d_S = \mathbf{E}[X_S] = w_S^{(1)} - \sum_{A \subseteq U} p_A \sum_{e \in S} z^*_{A,e}$ is, by lemma 4.8, a subgradient at x. As we have already proved, $X_S \in [-w_S^{(2)}, w_S^{(1)}]$ for each S, so, using lemma 4.10 with error probability δ/m and $c = \omega/2$, we can estimate the expectation $\mathbf{E}[X_S]$ by $\hat{d}_S$ using the claimed number of samples, so that for each S we have that $d_S - 2 \cdot \frac{\omega}{2} \cdot w_S^{(1)} \leq \hat{d}_S \leq d_S \Rightarrow d_S - \omega w_S^{(1)} \leq \hat{d}_S \leq d_S$ with probability at least δ/m. Thus, the error probability for each $\hat{d}_S$ is at most δ/m, and, so, the probability that at least one $\hat{d}_S$ is out of the expected interval is at most $m \cdot \delta/m = \delta$, that is, $\Pr[\forall S, d_S - \omega w_S^{(1)} \leq \hat{d}_S \leq d_S] \geq 1 - \delta$. So, by lemma 4.8, the vector $\hat{d} = \{\hat{d}_S\}$ is an ω-subgradient at x with probability at least $1 - \delta$.

□

Now, we go back to theorem 4.2 and use the time bound $T(\omega)$ to get:

Lemma 4.11. *Using the above procedure for computing ω-subgradients, procedure* `FindOpt` *finds a feasible solution $\bar{x}$ such that $h(\bar{x}) \leq OPT/(1-\gamma) + \varepsilon$ with probability at least $1 - \delta$ in time* $\text{poly}\big(\textit{input size}, \lambda, \frac{1}{\gamma}, \ln\left(\frac{1}{\varepsilon}\right), \ln\left(\frac{1}{\delta}\right)\big)$.

Proof. Theorem 4.2 gives the performance guarantee and accounts for the time taken excluding the time taken to compute an ω-subgradient. We now need to show that with high probability every vector calculated is an ω-subgradient for $\omega = \gamma/(2n)$. The total number of times that we need to calculate an ω-subgradient is at most $N+n$. So, setting the error probability to $\delta/(N+n)$ in corollary 4.2 we get that $T(\omega) = \frac{400\lambda^2 \cdot 4n^2}{3\gamma^2} \ln\left(\frac{m(N+n)}{\delta}\right) = O\big(\frac{\lambda^2 n^2}{\gamma^2} \ln\left(\frac{m(N+n)}{\delta}\right)\big)$ samples suffice to ensure that each individual vector computed is an ω-subgradient with probability at least $1 - \delta/(N+n)$. Since we compute at most $(N+n)$ subgradients, we get that the probability that at least one computation fails is at most $(N+n) \cdot \delta/(N+n) = \delta$. Thus, with probability at least $1 - \delta$ all the subgradients computed are indeed ω-subgradients. As for the total time needed, it is $O\left(T(\omega) \cdot m^2 \ln^2(\frac{KRm}{V\varepsilon})\right)$, which is polynomial in the input size, λ, $\frac{1}{\gamma}$, $\ln\left(\frac{1}{\varepsilon}\right)$ and $\ln\left(\frac{1}{\delta}\right)$.

□

All these lead to our main result.

Theorem 4.3. *Procedure* `ConvOpt` *computes a feasible solution to (SSC-P2) of cost at most $(1+k) \cdot OPT$ with probability at least $1 - 2\delta$ in time polynomial in the input size, λ, $\frac{1}{k}$ and $\ln\left(\frac{1}{\delta}\right)$.*

Proof. By lemma 4.6 we know that `ConvOpt` determines with probability at least $1-\delta$ whether $OPT \geq p/\lambda$, and in this case calls `FindOpt`. The multiplicative guarantee and the time bound now follow from lemma 4.11 by setting $\gamma = k/3$ and $\varepsilon = kp/(2\lambda)$ (this can be easily verified with simple calculations). The total error probability requires at least one error in one of the procedures `ConvOpt` and `FindOpt`. And, since both have error probability at most δ, the total error probability is at most 2δ.

□

We conclude this analysis by returning to our initial problem, the Stochastic Set Cover. Using the rounding theorem 4.1 and the algorithm described above, one gets this:

Theorem 4.4. *There exists a (randomized) $(2\ln n + \varepsilon)$-approximation algorithm for the 2-stage Stochastic Set Cover.*

4.3 A general class of two-stage stochastic programs

We now proceed to apply the algorithm `ConvOpt` to a wide class of 2-stage stochastic linear programs. The description that follows is the same as in §5 of [38]. Here x and r_A denote, as usual, the stage I decisions and the corresponding recourse actions in scenario A respectively. Additionally, the formulation below captures settings where one might need to take some extra decisions, which do not have corresponding first-stage actions, when a scenario A materializes; e.g., in facility location, we need to assign the (random) demands of clients in a scenario to facilities. We use variables s_A, and constraints specified by B^A, D^A, h^A, j^A to encode such decisions.

(Stoc-P):

$$\begin{aligned}
\text{min:} \quad & w^{(1)} \cdot x + f(x) \quad \text{subject to } x \in \mathcal{P} \subseteq \mathbb{R}^m_{\geq 0}, \\
\text{where} \quad & f(x) = \sum_{A \in \mathcal{A}} p_A f_A(x), \\
\text{and} \quad & f_A(x) = \text{min:} \quad w^A \cdot r_A + q^A \cdot s_A \\
& \qquad\qquad\quad \text{s.t:} \quad B^A s_A \geq h^A \\
& \qquad\qquad\qquad\qquad D^A s_A + T^A r_A \geq j^A - T^A x \\
& \qquad\qquad\qquad\qquad r_A, s_A \geq \mathbf{0}, r_A \in \mathbb{R}^m, s_A \in \mathbb{R}^q.
\end{aligned}$$

(It was later ([43]) observed that constraints of the form $B^A s_A \geq h^A$ are not needed, as they are equivalent to $\binom{B^A}{D^A} s_A + \binom{\mathbf{0}}{T^A} r_A \geq \binom{h^A}{j^A} - \binom{\mathbf{0}}{T^A} x$. We decided however to keep the original form, as stated in [38]).

Here $\mathcal{A}$ denotes the set of all possible scenarios, and $\mathcal{P}$ is the feasible region polytope. We require that (a) $T^A \geq \mathbf{0}$ for every scenario A, and (b) at every point $x \in \mathcal{P}, f(x) \geq 0$ and that the primal and dual problems corresponding to $f_A(x)$ be feasible for every scenario A. A sufficient condition for (b) is to insist that $0 \leq f_A(x) < +\infty$ at every point $x \in \mathcal{P}$ and every scenario $A \in \mathcal{A}$. We can relax condition (a) somewhat and solve a more general class of programs that allow one to incorporate upper bounds on the second-stage decisions r_A, under certain conditions. We assume that $x = \mathbf{0}$ lies in $\mathcal{P}$, since we would like to be able to express the option where one does nothing in the first stage and defers all decisions to stage II.

The essential property of this class of programs is that constraints $D^A s_A + T^A r_A \geq j^A - T^A x$ have the same matrix T^A multiplying the recourse vector r_A and the first-stage vector x in every scenario A. This implies that the stage I decisions given by the vector x and the stage II decisions for scenario A given by the vector r_A act in the same capacity. Observe that this class of stochastic programs is rich enough to model stochastic problems with scenario-dependent recourse (that is, stage II) costs. To prevent an exponential blowup in the input, we consider an oracle model where an oracle supplied with scenario A reveals the scenario-dependent data $(w^A, q^A, h^A, j^A, B^A, D^A, T^A)$; procedure `ConvOpt` will need to query this oracle only a polynomial number of times. We will see that the above class of programs can model various practical problems, such as covering problems (Vertex Cover, Set Cover) and facility location problems.

We will not go into any details about how we can use the algorithm `ConvOpt` to solve the above class of programs. The analysis is very similar to the one presented in the previous section regarding set cover, and can be found in [38]. We will only say that we again assume that $\lambda = \max\left(1, \max_{A \in \mathcal{A}, S} \frac{w_S^A}{w_S^{(1)}}\right)$ is known and also use the notion of non-null scenario with total cost at least 1. To be more precise, we define a null-scenario to be a scenario A for which $f_A(x)$ is minimized at $x = \mathbf{0}$, i.e., $f_A(\mathbf{0}) = \min_{x \in \mathcal{P}} f_A(x)$. We can again reach a similar result.

Theorem 4.5. *Procedure* `ConvOpt` *can be used to obtain a feasible solution to (Stoc-P) of cost at most $(1+k)\cdot OPT$ with probability at least $1-2\delta$, in time* $\texttt{poly}\big(\textit{input size}, \lambda, k, \ln\left(\frac{1}{\delta}\right)\big)$.

Note 4.2. *The above theorem can be extended to problems where the second stage scenario is specified by a parameter ξ that is continuously distributed with probability density function $p(\xi)$. In such problems all we have to do is replace the summations by integrals.*

Before moving on to some applications of the above theorem, we will state one last result. We will prove that a scheme with an approximation guarantee as the above has a running time that necessarily depends on the inflation factor.

4.4 The dependence of the running time on the inflation factor

Both the techniques presented in chapters 3 and 4 have running time depending on the (maximum) inflation factor of the problem, which is considered to be known to the algorithms. We will prove here that such a dependence is necessary, if we want to achieve a multiplicative guarantee on our approximation.

To prove this, we will consider a simple instance of Stochastic Set Cover with a single set and a single element. The example exposes the inherent limitation of using a black-box to infer knowledge about the probability distribution on scenarios; it is straightforward to generalize the example to construct lower bound instances for other stochastic problems.

Consider an instance of SSC with universe $U = \{e\}$ and one set $S = U$, where $w_S^{(1)} = 1$ and $w_S^{(2)} = \lambda$, and let p denote the probability that scenario $\{e\}$ occurs (this probability is unknown to our algorithm). The only decision that the algorithm has to make is whether to buy set S in stage I or wait until stage II where the actual data will be revealed. Let $\mathcal{A}_N$ denote an algorithm that draws exactly N samples. Let O^* denote the value of the integer optimum solution.

Theorem 4.6. *If $\mathcal{A}_N$ returns a (fractional) solution of cost at most $c \cdot O^*$ with probability at least $1-\delta$ where $1 \leq c < \frac{\lambda}{2}$, then it must be that $N \geq \lambda \ln(\frac{1}{\delta}-1)/(2c)$. The bound applies even if $\mathcal{A}_N$ returns only a fractional solution of cost at most $c \cdot O^*$.*

Proof. In order to prove the above theorem, we will choose the parameters of our problem in such a way so that any algorithm that tries to achieve approximation factor c with error probability at most δ, and which samples fewer times than the number stated in the theorem, fails. Let X be a random variable that denotes the number of times scenario $\{e\}$ occurs in the N samples. If $X = 0$, then $\mathcal{A}_N$ must choose to defer to stage II with probability at least $1-\delta$ (the algorithm may flip coins), that is, it must return the integer solution $x = 0$ with probability at least $1-\delta$. Otherwise, it will return a non-zero cost solution with probability at least δ, and in the case where $p = 0$ and, so, $O^* = 0$, $\mathcal{A}_N$ will pick a non-zero fraction of set S in stage I with probability at least δ and thus incur a non-zero cost, that is, a cost greater than $c \cdot O^*$, with probability at least δ.

Choose any $\epsilon > 0$ such that $c \leq \frac{\lambda}{2(1+\epsilon)}$ and consider any $\epsilon' > 0$ where $\epsilon' \leq \epsilon$. Set $p = (1+\epsilon')c/\lambda \leq \frac{1}{2}$ and define $N_0(\epsilon') = (\lambda \ln(\frac{1}{\delta}-1))/(2(1+\epsilon')c)$. Let $r = Pr[X=0] = (1-p)^N > e^{-2pN}$ (this can be easily derived through simple calculus and since $p \leq 1/2$). The optimal solution is to pick S in stage I, and incur a cost of 1, since the objective function is $h(x) = x + p\lambda r = x + (1+\epsilon')cr$, where $x + r \geq 1$. But if $N < N_0(\epsilon')$, then $r > e^{-2pN_0(\epsilon')} = \frac{\delta}{1-\delta}$, so with probability at least $(1-\delta)r > \delta$, $\mathcal{A}_N$ will choose the solution $x = 0$ and incur a cost of $(1+\epsilon')c > c \cdot O^*$, since $O^* = 1$. Therefore for $\mathcal{A}_N$ to satisfy the required performance guarantee we must have $N \geq N_0(\epsilon')$ for every $\epsilon' \in (0, \epsilon]$ which implies that $N \geq \lambda \ln(\frac{1}{\delta}-1)/(2c)$.

□

Corollary 4.3. *If algorithm $\mathcal{A}_N$ returns a (fractional) solution of expected cost at most $c \cdot O^*$ where $1 \leq c < \frac{\lambda}{6}$, then it must be that $N \geq (\lambda \ln 2)/(6c)$.*

Proof. Let $Y \geq 0$ denote the cost of the solution returned by $\mathcal{A}_N$. We have that $\mathbf{E}[Y] \leq c \cdot O^*$. By Markov's inequality, we get $\Pr[Y \geq 3c \cdot O^*] \leq \frac{\mathbf{E}[Y]}{3c \cdot O^*} \leq \frac{1}{3}$. Thus, $\mathcal{A}_N$ returns a solution of cost at most $3c \cdot O^*$ with probability at least $\frac{2}{3}$. The claim now follows from theorem 4.6 by substituting δ with $1/3$ and c with $3c$.

□

4.5 Applications

We will now present some problems that can be modeled in the form of (Stoc-P). The class of problems described by (Stoc-P) is, as already mentioned, quite general and can be applied to multicommodity flow problems, covering problems, facility location problems and others.

4.5.1 Vertex Cover

The 2-stage (weighted) Stochastic Vertex Cover is a special case of the 2-stage Stochastic Set Cover, which has already been analyzed. In general, Vertex Cover is the special case of Set Cover in which the universe is the set of edges and each set that can be bought corresponds to one vertex and includes all the edges that are adjacent to this vertex. In the case of the 2-stage Stochastic Vertex Cover, we do not know the edge set to be covered, but only a distribution over scenarios that are subsets of the edges.

The previous results known for this problem were an 8-approximation algorithm in the black-box model (see section 3.4.3), and a 3-approximation algorithm in the setting where each edge is independently activated; both results are based on the boosted sampling framework presented in chapter 3 (more details in [17]) and are obtained under the restrictions imposed by the framework. Ravi and Sinha ([35]) gave a 2-approximation algorithm when there are only polynomially many scenarios, but the second-stage costs may be scenario dependent.

Since the stochastic vertex cover problem is, as already mentioned, a special case of the stochastic set cover problem, and the deterministic vertex cover LP is known to have an integrality gap of 2, by corollary 4.1, we obtain, for any $\varepsilon > 0$, a $(4+\varepsilon)$-approximation algorithm for the stochastic version with black-box probability distributions and scenario-dependent second-stage costs. This improves the result of Gupta etc, which also has the restriction that the approximation factor regards the expected value returned, while with the techniques presented in this chapter we can make the error probability as low as possible.

Theorem 4.7. *For any $\varepsilon > 0$, there is a randomized $(4 + \varepsilon)$-approximation algorithm for the 2-stage Stochastic Vertex Cover Problem with arbitrary probability distributions and scenario-dependent stage II costs.*

4.5.2 The Metric Uncapacitated Facility Location Problem

The problem was defined in section 3.4.2. The difference now is that inflation factor can be different for each facility. We can open some facilities in stage I paying a cost of $f_i^{(1)}$ for opening facility i, then the actual scenario A with demands d_j^A is revealed, and we may choose to open some more facilities in stage II, incurring a cost of f_i^A for each facility i that we open in scenario A. As indicated by the notation, the recourse costs f_i^A may in general be scenario-dependent.

The deterministic facility location problem can be cast as an instance of the set cover problem where the sets corresponds to stars, each consisting of a facility and a set of clients assigned to it, and the elements to be covered are the clients. However the stochastic version is not a special case of the stochastic set cover problem. This is because in the stochastic problem, whereas we may decide to open some facilities in stage I, we are required to assign clients to these facilities only after the scenario is revealed. That is, we do not specify the coverage of a first-stage facility in stage I; in fact, this coverage will in general depend on the second-stage scenario. Thus, opening a facility in stage I does not quite correspond picking a star (which determines a fixed coverage) in stage I (rather one is picking a collection of stars), so the star-covering formulation does not apply to the stochastic generalization.

For the special case where $f_i^A = \lambda f_i^{(1)}$ for each $i \in \mathcal{F}$, Gupta et al. [17] gave an 8.45-approximation algorithm in the black box model (see section 3.5) and a 6-approximation algorithm in the setting where each client is activated independently. Ravi and Sinha [35] gave an LP-rounding based 8-approximation algorithm for the polynomial scenarios setting that can handle scenario-dependent facility opening and client assignment costs, where the assignment cost in scenario A is $c_{ij}^A = \gamma^A c_{ij}$ for all i, j. This was improved by Mahdian [28] to a factor of 3 via a primal-dual algorithm. However both of these algorithms need to explicitly know scenario-specific information, which renders them unsuitable when there are exponentially many scenarios. The rounding algorithm in [35] needs to know the optimal fractional solution for each stage II scenario, whereas the primal-dual algorithm in [28] requires explicit knowledge of p_A, f_i^A 's and d_j^A 's for every scenario A.

The work of Shmoys and Swamy generalizes or improves upon these results. They consider a convex programming relaxation of the problem and give a different rounding approach that decides which facilities to open in stage I based on only the stage I fractional solution. Combined with the algorithm to solve the convex program, this yields a 3.378-approximation algorithm in the black-box model with scenario-dependent costs.

Theorem 4.8. *There is a randomized* $(3.378 + \varepsilon)$*-approximation algorithm for SUFL based on rounding.*

Chapter 5

The Sample Average Approximation Method

5.1 Introduction

A natural approach in the problems we have already considered, where the number of scenarios is too large or infinite, is to take some number of samples from the distribution and solve the sampled problem that occurs. In such cases, we hope that for a suitably chosen sample size, a good solution to the sampled problem will also be a good solution to the initial problem. This approach is called the *Sample Average Approximation* method (SAA). The SAA method is an example of a scenario reduction technique, in that it replaces a complex distribution over a large (or even infinite) number of scenarios by a simpler, empirical distribution over some observed scenarios. Note that the SAA problem in any case remains a stochastic problem.

The SAA method is well known and falls under the broader area of Monte Carlo sampling. It is used in practice and has been extensively studied and analyzed in the stochastic programming literature. The interested reader can refer to [36] and [37].

Our goal here is to prove that a polynomial number of samples can give an $(1+\varepsilon)$-approximation to our problems. Such a result would mean that we can reduce our black-box problem to a polynomial-scenario problem. However, as we will see, even the SAA problem is usually hard to solve. Thus, in order to actually reach a useful result, we must show that we can in some way use an approximation algorithm for the SAA problem and maintain the approximation guarantee to the true problem.

We are going to present the approach of Charikar, Chekuri and Pál [7] who give a simple proof relying on Chernoff bounds. The proof shows that the SAA method works because of statistical properties of the objective function and its domain, and not on computational properties of optimizing the function in this domain. This approach works both for the discrete and the continuous case, in contrast to the approach of Swamy and Shmoys who reach the same results ([42], [43]) but only in continuous spaces. In any case, we are going to present as well the Swamy and Shmoys approach when we will talk about multistage stochastic problems (see section 9.3.2).

We should mention here that these two works are the first to give a sample bound polynomial in the input size, the maximum inflation factor λ, $\log(1/\delta)$ and $1/\varepsilon$ (δ is the error probability). Before that, the best result was due to [24] and gave a bound that depended on the variance of a certain quantity that need not depend polynomially on the input size or λ. Another remark here is that in any case, we cannot estimate the value $f(x^*)$ of our objective value, but only a solution $\bar{x}$ such that the value $f(\bar{x})$ is near to $f(x^*)$. To get a $(1+\varepsilon)$-approximation for the value $f(x^*)$, the aforementioned dependence on the variance of a certain random variable of our problem cannot be avoided.

We are now ready to proceed to our analysis. We remind the reader that we work in the black-box model. We will use the notation used in [7]. The problems considered are of the form

$$\min_{x \in X} f(x) = c(x) + \mathbf{E}_\omega[q(x, \omega)]$$

where f is the objective value function that is written as a function of the first-stage decision vector $x \in X$, and ω is a scenario. The first-stage cost is given by $c : X \to \mathbb{R}_{\geq 0}$ and the cost of the second stage in a particular scenario ω, given a first-stage vector x is the optimum of the second-stage minimization problem, that is:

$$q(x, \omega) = \min_{r \in R}\{cost_\omega(r) \mid (x, r) \text{ is a feasible solution for scenario } \omega\}$$

where R is the set of recourse actions. By taking now N independent samples $\omega_1, \omega_2, ..., \omega_N$ we can define the sample average function as:

$$\hat{f}(x) = c(x) + \frac{1}{N}\sum_{i=1}^{N} q(x, \omega_i)$$

We consider stochastic two stage problems that satisfy the following properties:

- *Non-negativity*: the functions $c(x)$ and $q(x, \omega)$ are non-negative for every first stage action x and every scenario ω.

- *Empty first stage*: we assume that there is an empty first-stage action, $0 \in X$. The empty action incurs no first-stage cost, i.e. $c(0) = 0$, but is least helpful in the second stage. That is, for every $x \in X$ and every scenario ω, $q(x, \omega) \leq q(0, \omega)$.

- *Bounded inflation factor*: we present the definition of the inflation factor that is given in [7], that is, $\lambda \geq 1$ is the least number such that for every scenario ω and every $x \in X$, we have $q(0, \omega) - q(x, \omega) \leq \lambda c(x)$.

The above definition of the inflation factor is quite general and captures the cases we have already studied. Intuitively, it means that the profit that we can make by taking some first-stage actions in any scenario is bounded by the cost of these first-stage actions multiplied by the inflation factor, that is, profit is bounded because the costs are bounded by inflation.

We have already used X to denote the set of first-stage decisions. Let $\mathcal{X}$ be the set of elements we can buy, so that $X = \{0, 1\}^{\mathcal{X}}$. We will prove that $\lambda = \max_{\omega, y \in \mathcal{X}} \frac{cost_\omega(y)}{c(y)} \Rightarrow q(0, \omega) - q(x, \omega) \leq \lambda c(x)$.

We have that $cost_\omega(y) \leq \lambda c(y)$, $\forall \omega, y \in \mathcal{X}$. Let F_0 be the set of recourse actions for a scenario ω when the first-stage is empty, which means that $q(0, \omega) = cost_\omega(F_0)$. Now let x be any first-stage vector (translating to set F_x), and r be the corresponding second-stage recourse (translating to set F_r). We have that $(F_x \cup F_r) \in Sols(\omega) \Rightarrow q(0, \omega) \leq cost_\omega(F_x) + cost_\omega(F_r)$. Observe that $cost_\omega(F_r) = q(x, \omega)$ and $cost_\omega(F_x) \leq \lambda c(x)$. Thus, we get that $q(0, \omega) \leq \lambda c(x) + q(x, \omega) \Rightarrow q(0, \omega) - q(x, \omega) \leq \lambda c(x)$.

5.2 The Discrete Case

We are here going to discuss the case where the first stage decision x ranges over a finite set of choices X (that is the case for most combinatorial problems). Let x^* denote an optimal solution to the initial problem and Z^* its value $f(x^*)$. We also assume that ε is small, say $\varepsilon < 0.1$. Our main result is:

Theorem 5.1. *Any exact minimizer $\bar{x}$ of the function $\hat{f}(.)$ constructed with $\Theta(\lambda^2 \frac{1}{\varepsilon^4} \ln |X| \ln \frac{1}{\delta})$ samples is, with probability $1 - 2\delta$, a $(1 + O(\varepsilon))$-approximate minimizer of the function $f(.)$.*

The notion of exact and approximate minimizers is clarified below. The above theorem states that if we can solve exactly the SAA problem, then we have a near-optimal solution to the true problem. However, note that there is no guarantee that an approximate solution to the SAA problem will be an approximate solution to the true problem. And, since the SAA problems are usually hard, the above theorem cannot be utilized in most cases. We will show that there are techniques to get such a result for approximate minimizers of the SAA function, too.

Definition 5.1. An $x^* \in X$ is said to be an exact minimizer of the function $f(.)$ if for all $x \in X$ it holds that $f(x^*) \leq f(x)$. An $\bar{x} \in X$ is an α-approximate minimizer of the function $f(.)$ if for all $x \in X$ it holds that $f(\bar{x}) \leq \alpha f(x)$.

The main tool that we will use in our analysis is the Chernoff bound. We will be using the following version of the bound: (see [32] or [20] for more details)

Lemma 5.1 (Chernoff bound)**.** *Let $X_1, ..., X_N$ be independent random variables with $X_i \in [0, 1]$ and let $X = \sum_{i=1}^{N} X_i$. Then, for any $\varepsilon \geq 0$, we have $\Pr[X - \mathbf{E}[X] > \varepsilon N] \leq e^{-\varepsilon^2 N}$. We also get $\Pr[\, |X - \mathbf{E}[X]| > \varepsilon N \,] \leq 2e^{-\varepsilon^2 N}$.*

The first and most natural approach towards the proof of theorem 5.1 is to try and show that if N is large enough, then the functions f and $\hat{f}$ will be close to each other, in that with high probability $|f(x) - \hat{f}(x)| \leq \varepsilon f(x)$. Unfortunately, this is not the case, as for any particular x the random variable $q(x, \omega)$ may have very high variance. However, intuitively, one can think that the high variance of $q(x, \omega)$ is caused by a few "bad" scenarios of very high cost but low probability, and whose cost is not very sensitive to the first-stage vector x. We will indeed prove that this is the case, and so these scenarios do not affect the choice of x significantly.

To formalize this idea, we divide the scenarios into two classes. We call a scenario ω high, if its second stage cost $q(0, \omega)$ exceeds a threshold M, and low otherwise. We will see that $M = \lambda Z^*/\varepsilon$ works conveniently.

As already mentioned, we approximate function f by taking N independent samples $\omega_1, ..., \omega_N$. We define the following two functions to account for the contributions of low and high scenarios respectively.

$$\hat{f}_l(x) = \frac{1}{N} \sum_{i:\, \omega_i \text{ is low}} q(x, \omega_i)$$

and

$$\hat{f}_h(x) = \frac{1}{N} \sum_{i:\, \omega_i \text{ is high}} q(x, \omega_i).$$

Note that $\hat{f}(x) = c(x) + \hat{f}_l(x) + \hat{f}_h(x)$. We make a similar definition for the initial function $f(.)$. Let $p = \Pr_\omega[\omega \text{ is a high scenario}]$. Then,

$$f_l(x) = (1 - p) \cdot \mathbf{E}[q(x, \omega) \,|\, \omega \text{ is low}] \quad \text{and} \quad f_h(x) = p \cdot \mathbf{E}[q(x, \omega) \,|\, \omega \text{ is high}]$$

so that $f(x) = c(x) + f_l(x) + f_h(x)$.

We are now going to bound the probability p.

Lemma 5.2. *The probability mass p of high scenarios is at most $\frac{\varepsilon}{(1-\varepsilon)\lambda}$.*

Proof. Observe that for a high scenario ω we have that $q(0,\omega)-q(x^*,\omega) \leq \lambda c(x^*)$ and $q(0,\omega) \geq M$, so we get that $q(x^*,\omega) \geq M - \lambda c(x^*)$. Also, $Z^* \geq f_h(x^*) = p \cdot \mathbf{E}[q(x^*,\omega) \,|\, \omega \text{ is high}]$, and, since $q(x^*,\omega) \geq M - \lambda c(x^*)$ for every high scenario ω, we get that $\mathbf{E}[q(x^*,\omega) \,|\, \omega \text{ is high}] \geq M - \lambda c(x^*)$. Thus, $Z^* \geq p \cdot (M - \lambda c(x^*))$. Substituting $M = \lambda Z^*/\varepsilon$ and using the fact that $c(x^*) \leq Z^*$ we obtain

$$Z^* \geq Z^* \big(\frac{1}{\varepsilon} - 1\big)\lambda p.$$

So,

$$\big(\frac{1}{\varepsilon} - 1\big)\lambda p \leq 1 \Rightarrow p \leq \frac{\varepsilon}{(1-\varepsilon)\lambda}.$$

□

In order to prove theorem 5.1 we show that each of the following properties hold with probability at least $1-\delta$:

(P1) $\forall x \in X,\ \ |f_l(x) - \hat{f}_l(x)| \leq \varepsilon Z^*.$

(P2) $\forall x \in X,\ \ \hat{f}_h(0) - \hat{f}_h(x) \leq 2\varepsilon c(x).$

(P3) $\forall x \in X,\ \ f_h(0) - f_h(x) \leq 2\varepsilon c(x).$ (in fact, this property holds with probability 1)

We will now prove theorem 5.1 using the above properties.

***Proof of theorem* 5.1.** Since the error probability of each of the properties (P1) and (P2) is at most δ, we can assume that with probability at least $1-2\delta$ all three properties hold. For any $x \in X$ we have

$$\begin{aligned}
f_l(x) &\leq \hat{f}_l(x) + \varepsilon Z^* && (P1)\\
f_h(x) &\leq f_h(0) && (\text{since } \forall\omega,\ q(x,\omega) \leq q(0,\omega))\\
0 &\leq \hat{f}_h(x) + 2\varepsilon c(x) - \hat{f}_h(0) && (P2)
\end{aligned}$$

Adding the above inequalities we get

$$\begin{aligned}
(f_l(x) + f_h(x)) - (\hat{f}_l(x) + \hat{f}_h(x)) &\leq \varepsilon Z^* + 2\varepsilon c(x) + f_h(0) - \hat{f}_h(0) \Rightarrow\\
f(x) - \hat{f}(x) &\leq \varepsilon Z^* + 2\varepsilon c(x) + f_h(0) - \hat{f}_h(0)
\end{aligned} \tag{5.1}$$

By a similar reasoning and by using the inequalities

$$\begin{aligned}
\hat{f}_l(x) &\leq f_l(x) + \varepsilon Z^* && (P1)\\
\hat{f}_h(x) &\leq \hat{f}_h(0)\\
0 &\leq f_h(x) + 2\varepsilon c(x) - f_h(0) && (P3)
\end{aligned}$$

we get

$$\hat{f}(x) - f(x) \leq \varepsilon Z^* + 2\varepsilon c(x) + \hat{f}_h(0) - f_h(0) \tag{5.2}$$

Now, let x^* and $\bar{x}$ be minimizers of the functions $f(.)$ and $\hat{f}(.)$ respectively. Setting $x = \bar{x}$ in 5.1 and $x = x^*$ in 5.2 and adding them up we get

$$f(\bar{x}) - \hat{f}(\bar{x}) + \hat{f}(x^*) - 2\varepsilon c(\bar{x}) \leq f(x^*) + 2\varepsilon Z^* + 2\varepsilon c(x^*).$$

Since $\hat{f}(\bar{x}) \leq \hat{f}(x^*)$, we get

$$f(\bar{x}) - 2\varepsilon c(\bar{x}) \leq f(x^*) + 2\varepsilon Z^* + 2\varepsilon c(x^*).$$

Note now that $c(x) \leq f(x)$, $\forall x \in X$, and $Z^* = f(x^*)$. Thus, we get

$$\begin{aligned}
(1-2\varepsilon) f(\bar{x}) &\leq (1+4\varepsilon) f(x^*) \Rightarrow \\
f(\bar{x}) &\leq \frac{1+4\varepsilon}{1-2\varepsilon} f(x^*) \Rightarrow \\
f(\bar{x}) &\leq \Big(1 + \frac{6\varepsilon}{1-2\varepsilon}\Big) f(x^*) \Rightarrow \\
f(\bar{x}) &\leq (1+O(\varepsilon)) f(x^*)
\end{aligned}$$

□

We now proceed to prove the above properties **(P1 - P3)**. We will make repeated use of the Chernoff bound stated in lemma 5.1. Properties **(P2)** and **(P3)** are an easy corollary of the *bounded inflation factor* property once we realize that the probability of drawing a high sample from the distribution π is small; and that the fraction of high samples we draw will be small as well with high probability. The reader should notice here the similarity in the results obtained for the functions f_h and $\hat{f}_h$. Let N_h denote the number of high samples in $\omega_1, ..., \omega_N$.

Lemma 5.3. *With probability* $1-\delta$, $N_h/N \leq 2\varepsilon/\lambda$.

Proof. Let X_i be the indicator variable that is equal to 1 if the sample ω_i is high and 0 otherwise. Then, $N_h = \sum_{i=1}^N X_i$ is a sum of i.i.d. 0-1 variables and $\mathbf{E}[N_h] = \sum_{i=1}^N \mathbf{E}[X_i] = Np$. We are going to use Chernoff bounds to bound the probability of the event $N_h/N \leq 2\varepsilon/\lambda$. We have that

$$\begin{aligned}
N_h/N &> 2\varepsilon/\lambda \Rightarrow \\
N_h &> 2\varepsilon N/\lambda \Rightarrow \\
N_h - \mathbf{E}[N_h] &> 2\varepsilon N/\lambda - pN \Rightarrow \text{(using lemma 5.2)} \\
N_h - \mathbf{E}[N_h] &> 2\varepsilon N/\lambda - N\frac{\varepsilon}{(1-\varepsilon)\lambda}
\end{aligned}$$

Using Chernoff bounds (lemma 5.1) we get

$$\Pr\Big[N_h - \mathbf{E}[N_h] > \frac{\varepsilon}{\lambda} N\big(2 - \frac{1}{1-\varepsilon}\big)\Big] \leq \exp\Big(-\frac{\varepsilon^2}{\lambda^2}\frac{(1-2\varepsilon)^2}{(1-\varepsilon)^2} N\Big).$$

With $\varepsilon < 1/3$ and N chosen as in theorem 5.1, this probability is at most δ. Thus, with probability at least $1-\delta$ we have that $N_h/N \leq 2\varepsilon/\lambda$ (simple calculus confirms the result).

□

We can now prove properties **(P2-P3)**.

Corollary 5.1. *Property* (**P2**) *holds with probability* $1-\delta$, *and property* (**P3**) *holds with probability 1.*

Proof. By lemma 5.3 we have that $N_h \leq 2N\varepsilon/\lambda$, with probability at least $1-\delta$ and $\varepsilon < 1/3$. Then we have

$$\begin{aligned}
\hat{f}_h(0) - \hat{f}_h(x) &= \frac{1}{N} \sum_{i:\,\omega_i \text{ is high}} (q(0,\omega_i) - q(x,\omega_i)) \\
&\leq \frac{N_h}{N} \lambda c(x) \\
&\leq \frac{2\varepsilon}{\lambda} \lambda c(x) \\
&= 2\varepsilon c(x),
\end{aligned}$$

by using the fact that $q(0,\omega) - q(x,\omega) \le \lambda c(x)$. This proves property **(P2)**.

Property **(P3)** can be proved in a similar way. We have

$$\begin{aligned} f_h(0) - f_h(x) &= p \cdot \mathbf{E}[q(0,\omega) \,|\, \omega \text{ is high}] - p \cdot \mathbf{E}[q(x,\omega) \,|\, \omega \text{ is high}] \\ &= p \cdot \mathbf{E}[q(0,\omega) - q(x,\omega) \,|\, \omega \text{ is high}] \\ &\le p\lambda \cdot \mathbf{E}[c(x) \,|\, \omega \text{ is high}] \\ &= p\lambda c(x) && \text{(using lemma 5.2)} \\ &\le \frac{\varepsilon}{1-\varepsilon} c(x) \\ &\le 2\varepsilon c(x) && \text{(since } \varepsilon < 1/2) \end{aligned}$$

And, since all results used are true with probability 1, then property **(P3)** is also true with probability 1.

□

We are now ready to prove that property **(P1)** holds with probability $1-\delta$. The following proof is the only place where we use the fact that X is finite. In the next section we will see that property **(P1)** holds even when $X \subseteq \mathbb{R}^n$, under an assumption that the function $c(\cdot)$ is linear and that $q(\cdot,\cdot)$ satisfies some certain Lipschitz-type property.

Lemma 5.4. *With probability at least $1-\delta$ it holds that for all $x \in X$,*

$$|f_l(x) - \hat{f}_l(x)| \le \varepsilon Z^*.$$

Proof. We are again going to use Chernoff bounds. Consider at first that the first-stage vector x is fixed. We can view $\hat{f}_l(x)$ as the arithmetic mean of N independent copies $Q_1, ..., Q_N$ of the random variable Q which is defined as

$$Q = \begin{cases} q(x,\omega) & \text{if } \omega \text{ is low} \\ 0 & \text{if } \omega \text{ is high} \end{cases}$$

Observe that $f_l(x) = \mathbf{E}[Q]$. Let Y_i be the variable Q_i/M and $Y = \sum_{i=1}^N Y_i$. Note that $Y_i \in [0,1]$ and $\mathbf{E}[Y] = \frac{N}{M} f_l(x)$. We now have

$$\begin{aligned} |f_l(x) - \hat{f}_l(x)| &= \left| \mathbf{E}[Q] - \frac{1}{N}\sum_{i=1}^N Q_i \right| \\ &= M\left| \frac{\mathbf{E}[Q]}{M} - \frac{1}{N}\sum_{i=1}^N \frac{Q_i}{M} \right| \\ &= M\left| \frac{\mathbf{E}[Q]}{M} - \frac{Y}{N} \right| \\ &= \frac{M}{N}\left| Y - \frac{N}{M}\mathbf{E}[Q] \right| \\ &= \frac{M}{N}|Y - \mathbf{E}[Y]| \end{aligned}$$

We want to calculate the probability of the event $|f_l(x) - \hat{f}_l(x)| > \varepsilon Z^*$. We have

$$\begin{aligned} |f_l(x) - \hat{f}_l(x)| > \varepsilon Z^* &\Leftrightarrow \\ \frac{M}{N}|Y - \mathbf{E}[Y]| > \varepsilon Z^* &\Leftrightarrow \\ |Y - \mathbf{E}[Y]| > \frac{N}{M}\varepsilon Z^* &\Leftrightarrow \\ |Y - \mathbf{E}[Y]| > \frac{N}{\lambda}\varepsilon^2 & \end{aligned}$$

Using now the Chernoff bound of lemma 5.1 we get

$$\Pr\left[|Y - \mathbf{E}[Y]| > \frac{N}{\lambda}\varepsilon^2\right] \leq 2\exp\Big(-\frac{\varepsilon^4}{\lambda^2}N\Big).$$

With N as in theorem 5.1, this probability is at most $\delta/|X|$ (with δ sufficiently small). Taking now the union bound over all $x \in X$, we obtain the desired claim.

□

5.3 Approximation algorithms and SAA

In many cases, as already stated, finding an exact minimizer of the sample function $\hat{f}(.)$, which is needed in order to apply theorem 5.1, is computationally hard. In such cases we need to use an approximate minimizer of the function $\hat{f}(.)$ which can be obtained through an approximation algorithm.

We will now explore the performance of the SAA method, as already presented, used with an α-approximation algorithm for minimizing the sample function $\hat{f}$. The lemma proved below is an adaptation of theorem 5.1.

Lemma 5.5. *Let $\bar{x}$ be an α-approximate minimizer for $\hat{f}$. Then, with probability at least $(1-2\delta)$,*

$$f(\bar{x})(1-2\varepsilon) \leq (1+4\varepsilon)\alpha f(x^*) + (\alpha - 1)(\hat{f}_h(0) - f_h(0))$$

We use a similar argument to the one used in the proof of theorem 5.1 in order to prove the above lemma. Its proof can be found in appendix C. We can see that $f(\bar{x})$ is a good approximation to $f(x^*)$ if $\hat{f}_h(0) - f_h(0)$ is small. By using Markov's Inequality, we get that $\Pr[\,|\hat{f}_h(x)| \geq (1+\frac{1}{k})f_h(x)\,] \leq \frac{\mathbf{E}[\hat{f}_h(x)]}{(1+\frac{1}{k})f_h(x)} = \frac{f_h(x)}{(1+\frac{1}{k})f_h(x)} = \frac{k}{k+1}$. Thus, $\Pr[\,\hat{f}_h(0) \leq (1+\frac{1}{k})f_h(0)\,] \geq 1 - \frac{k}{k+1} = \frac{1}{k+1}$, and so if we want to achieve multiplicative error $(1+\varepsilon)$, we must be content with probability of success only proportional to $1/\varepsilon$, and, in fact, we can easily construct distributions where the Markov bound is tight.

In order to improve our probability of success, there are two alternative solutions. We can boost our probability of success by repeating the sampling procedure and taking the "best" sample function that occurs. The other solution is to try and ignore the high cost samples. This does not affect significantly the quality of any solution while reducing the variance in evaluating the objective function. We present the two results, omitting the proofs. These results essentially mean that we have managed to reduce the black-box problems that we study to polynomial-scenario problems.

5.3.1 Approximation algorithms and repeating SAA

Theorem 5.2. *Consider a collection of k functions $\hat{f}^1, \hat{f}^2, \ldots \hat{f}^k$, such that $k = \Theta(\varepsilon^{-1}\ln\delta^{-1})$, and the $\hat{f}^i$ are independent sample average approximations of the function f, using $N = \Theta(\lambda^2\varepsilon^{-4}\cdot k\cdot \ln|X|\ln\delta^{-1})$ samples each. For $i = 1,\ldots,k$, let $\bar{x}^i$ be an α-approximate minimizer of the function $\hat{f}^i$. Let $i = arg\min_j \hat{f}^j(\bar{x}^j)$. Then, with probability $1-3\delta$, $\bar{x}^i$ is an $(1+O(\varepsilon))\alpha$-approximate minimizer of the function $f(.)$.*

The proof is quite straightforward and is based on the bounds obtained by Markov inequality, the independence of the sampling events and lemma 5.5.

5.3.2 Approximation algorithms and Sampling with Rejection

Instead of repeating the SAA method, we can use it only once and ignore the high cost samples. To make this idea more formal we will need the following lemma:

Lemma 5.6. *Let $g : X \to \mathbb{R}$ be a function satisfying $|f_l(x) + c(x) - g(x)| = O(\varepsilon)Z^*$ for every $x \in X$. Then, any α-approximate minimizer $\bar{x}$ of the function $g(.)$ is also an $\alpha(1 + O(\varepsilon))$-approximate minimizer of the function $f(.)$.*

Using the above lemma and the fact that with high probability we can bound the portion of the high cost samples in our total sample (as also stated in lemma 5.3), we reach our main result.

Theorem 5.3. *Let $\omega_1, \omega_2, ..., \omega_N$ be independent samples with $N = \Theta(\lambda^2\varepsilon^{-4} \cdot \ln|X| \ln \delta^{-1})$. Let $\omega'_1, \omega'_2, ..., \omega'_N$ be a reordering of the samples such that $q(0, \omega'_1) \le q(0, \omega'_2) \le ... \le q(0, \omega'_N)$. Then any α-approximate minimizer $\bar{x}$ of the function $\bar{f}(x) = c(x) + \frac{1}{N}\sum_{i=1}^{N'} q(x, \omega'_i)$, with $N' = (1 - 2\varepsilon/\lambda)N$ is a $(1 + O(\varepsilon))\alpha$-approximate minimizer of $f(.)$.*

We must highlight the fact that in many situations, computing $q(x, \omega)$ (or even $(q(0, \omega))$ requires us to solve an NP-hard problem. This makes the ordering that the above theorem states impossible. However, if we have an approximation algorithm with ratio β for computing $q(\cdot, \cdot)$, we can use it to order the samples instead. For the above theorem to work with such an approximation algorithm, the number of samples N needs to increase by a factor of β^2 and N' needs to be $(1 - 2\varepsilon/(\beta\lambda))N$.

5.4 From the Discrete to the Continuous

So far we have assumed that X, the set of first-stage decisions, is a finite set. We will now show that the above results can be extended in situations where $X \subseteq \mathbb{R}^n$. Although our goal in this thesis is algorithms for combinatorial problems, we present this extension so that the reader can make the analogy with chapter 4, where we actually go the other way, that is, from continuous to discrete.

Since all theorems depend on the validity of the three properties **(P1 - P3)**, we only need to prove that these three properties still hold when $X \subseteq \mathbb{R}^n$. Observe that in the proofs of properties **(P2)** and **(P3)** we did not use the fact that X was finite. Thus, they still hold for any X. So, the only thing remaining to do is prove that **(P1)** also holds.

In order to do so, we need to make some assumptions about our functions. These assumptions are reasonable, and are very similar to the ones made by Swamy and Shmoys in chapter 4. We first assume that $X \subseteq \mathbb{R}^n_{\ge 0}$. We will need the fact that the first-stage cost function is linear, that is, $c(x) = c^T \cdot x$, for some real vector $c = (c_1, ..., c_n)$ with non-negative coefficients. We are now going to introduce a Lipschitz-type property, which captures the notion of bounded variation that we will need, as in chapter 4.

Definition 5.2. The recourse function $q(\cdot, \cdot)$ is (λ, c)-Lipschitz, if for every scenario ω we have:

$$|q(x, \omega) - q(x', \omega)| \le \lambda \sum_{i=1}^{n} c_i |x_i - x'_i|.$$

Note that any (λ, c)-Lipschitz recourse function q satisfies $q(0, \omega) - q(x, \omega) \le \lambda c(x)$ (this is not always true if $c(.)$ is non-linear). We assume that the recourse functions involved in our problems are (λ, c)-Lipschitz, where c is the first-stage cost-vector.

We are going to use a standard meshing argument: if two functions $\hat{f}$ and f do not differ by much on a dense enough finite mesh, because of bounded gradient, they must approximately agree in the whole region covered by the mesh. This idea has been used in the context of stochastic optimization by various authors. Swamy and Shmoys have used a similar argument ([42] and [43]), which we will present in chapter 9.

We will use an n-dimensional grid of points with $\varepsilon/(n\alpha\lambda c_i)$ spacing in each dimension $1 \leq i \leq n$. To bound this grid, we observe that the i^{th} coordinate of any α-approximate minimizer $\bar{x}$ of f cannot be larger than $\alpha Z^*/c_i$, as otherwise we would have $f(\bar{x}) > c_i \cdot \alpha Z^*/c_i = \alpha Z^*$. So, we can assume that the feasible region lies within the box $0 \leq x_i \leq \alpha Z^*/c_i,\ \ 1 \leq i \leq n$. Thus, our mesh is actually

$$X' = \left\{ \left(i_1 \frac{\varepsilon Z^*}{n\lambda c_1}, ..., i_n \frac{\varepsilon Z^*}{n\lambda c_n} \right) \middle| (i_1, ..., i_n) \in \{0, 1, ..., \lceil n\alpha\lambda/\varepsilon \rceil\}^n \right\}$$

We are now going to prove an analog of lemma 5.4 for continuous sets.

Lemma 5.7. *If $N \geq \Theta(\lambda^2 \frac{1}{\varepsilon^4} n \ln(n\lambda/\varepsilon) \ln \delta)$, then with probability at least $1 - \delta$ we have that $|\hat{f}_l(x) - f_l(x)| \leq 3\varepsilon Z^*$ for every $x \in X$.*

Proof. In order to prove the validity of the sample size, we only have to observe that the size of X' is $(1 + \lceil n\alpha\lambda/\varepsilon \rceil)^n$, and so $\ln |X'| = O(n \ln(n\lambda/\varepsilon))$. Thus, by using lemma 5.4 we get that with probability at least $1 - \delta$, $|\hat{f}_l(x') - f_l(x')| \leq \varepsilon Z^*$ for every $x' \in X'$.

Consider now any point $x \in X$. From the construction of the grid, we get that there is a nearby mesh point $x' \in X'$ such that $\sum_{i=1}^n c_i |x_i - x'_i| \leq \sum_{i=1}^n c_i \frac{\varepsilon Z^*}{n\lambda c_i} = \varepsilon Z^*/\lambda$. Using the Lipschitz property we also get that $|f_l(x) - f_l(x')| \leq \varepsilon Z^*$ and $|\hat{f}_l(x) - \hat{f}_l(x')| \leq \varepsilon Z^*$. By triangle inequality we now get

$$\begin{aligned} |\hat{f}_l(x) - f_l(x)| &= |\hat{f}_l(x) - \hat{f}_l(x') + \hat{f}_l(x') - f_l(x') + f_l(x') - f_l(x)| \\ &\leq |\hat{f}_l(x) - \hat{f}_l(x')| + |\hat{f}_l(x') - f_l(x')| + |f_l(x') - f_l(x)| \\ &\leq 3\varepsilon Z^*. \end{aligned}$$

□

Using the above results, one can now reach a similar theorem to theorem 5.1, for continuous sets X.

5.5 Applications

The above results show that we can actually reduce black-box problems to polynomial-scenario problems. Thus, we can use results from the polynomial-scenario model, which in many cases are almost tight, i.e. the same as the deterministic counterparts, to get approximation algorithms for various problems in the black-box model.

5.5.1 Set Cover

Using the $O(\log n)$-approximation algorithm proposed by Ravi and Sinha in [35], we can get the following result.

Theorem 5.4. *There exists a randomized $O(\log n)$-approximation algorithm for the 2-stage Stochastic Set Cover with scenario-dependent costs and bounded inflation factor.*

5.5.2 The Metric Uncapacitated Facility Location Problem

As already mentioned in section 4.5.2, Mahdian [28] gave a 3-approximation to (SMUFLP) via a primal-dual algorithm. Thus, using this algorithm we can get the following result, which slightly improves the result of Shmoys, Swamy (see theorem 4.8).

Theorem 5.5. *There exists a randomized $(3 + \varepsilon)$-approximation algorithm for the 2-stage SUFLP with scenario-dependent costs and bounded inflation factor.*

5.5.3 Vertex Cover

Using the 2-approximation algorithm proposed by Ravi and Sinha in [35], we can get the following result, which improves the result of Shmoys and Swamy (see theorem 4.7).

Theorem 5.6. *There exists a randomized $(2 + \varepsilon)$-approximation algorithm for the 2-stage Stochastic Vertex Cover with scenario-dependent costs and bounded inflation factor.*

Chapter 6

Complexity of 2-stage Stochastic Programs

6.1 Introduction

We conclude our overview of 2-stage problems by examining the complexity of some models of 2-stage stochastic programs. We believe the reader is now in a position to understand the inherent difficulties of stochastic optimization, and be convinced that these problems are indeed hard. However, not much work has been done in the direction of proving theoretically this common belief. We will here present the only work that has been done in the field of computational complexity regarding stochastic programs. Dyer and Stougie in [13] prove that certain classes of 2-stage stochastic programs are ♯P-hard, and we are going through their ideas in this chapter.

6.2 A class of 2-stage programs

We will consider programs similar to the ones used in chapter 4. We will use the notation used in [13] so we will talk about maximization problems here (the same apply to minimization problems). The general formulation is the following:

$$\begin{aligned} \text{max:} \quad & c \cdot x + Q(x) \\ \text{s.t.:} \quad & Ax \leq b \\ & x \in X \subset \mathbb{R}^n_{\geq 0} \end{aligned}$$

with

$$Q(x) = \mathbf{E}[\, \max\{\mathbf{q} \cdot y \,|\, Wy \leq \mathbf{h} - \mathbf{T}x, y \in Y \subset \mathbb{R}^{n_1}_{\geq 0}\}\,]$$

Boldface characters are used to indicate random variables. Observe that the above formulation is quite general and captures many of the problems already discussed. Introducing the idea of scenarios, we get the *deterministic equivalent problem*, as it is often called, which is the usual formulation we have encountered where we write the expectation of the second stage as an explicit sum of scenarios and the corresponding probabilities. Each scenario is described by a triple of random variables $(\mathbf{q}, \mathbf{T}, \mathbf{h})$ and there are K scenarios, $(q^1, T^1, h^1), ..., (q^K, T^K, h^K)$, in total, each one having probability of occurrence p^i. Our problem can now be formulated as:

$$\begin{aligned} \text{max:} \quad & c \cdot x + \sum_{k=1}^{K} p^k (q^k \cdot y^k) \\ \text{s.t.:} \quad & Ax \leq b \\ & T^k x + W y^k \leq h^k, \quad k = 1, ..., K \end{aligned}$$

A crucial point in examining the complexity of stochastic programs is how the random parameters are described. If, as input of the problem, each scenario and its corresponding probability is specified completely, then the problem is polynomially solvable in case the decision variables have a convex feasible region (however, the reader should be careful whether polynomial in the input size means efficient, in cases where we have an exponential number of scenarios), and NP-complete if there are integrality constraints on the decision variables.

However, consider another extreme in which all parameters are independent identically distributed random variables, each having a value α_1 with probability p and α_2 with probability $1-p$. In that case, using m_1 for the number of rows of the T-matrices, there are $K = 2^{n_1+m_1n+m_1}$ possible scenarios, but they can be encoded with a relatively small number of bits. The size of the deterministic equivalent problem is exponential in the size of the input, and the complexity changes correspondingly. As already mentioned in the previous chapters, most of stochastic programming research focuses on methods to overcome this curse of problem size, which is usually caused by specification of the scenarios as combinations of realizations of independent random parameters. For example, the black-box model that we have used in conjunction with the sample average approximation method does not require a full listing of all scenarios.

From now on we will consider models wherein the random parameters are independently distributed. Of course, one can say that we are restricting ourselves to a small portion of all possible distributions, but a hardness result for such distributions gives an indication of the hardness for arbitrary distributions.

6.3 Complexity of 2-stage programs

We will focus on models with discretely distributed random parameters, as throughout this thesis we are mostly interested in combinatorial problems. The interested reader can refer to [13] for details about the treatment of continuously distributed parameters. We will establish ♯P-hardness of the evaluation of the second-stage expected value function $Q(x)$ for fixed x of a two-stage stochastic programming problem with discretely distributed parameters using a reduction from the problem *Graph Reliability*, ♯P-completeness of which has been proved in [46].

But let us first explain what the class ♯P is. A counting problem is a problem where we are interested in the number of solutions rather than in finding a specific solution. For example, the counting version of the *Hamilton Path* problem asks for the number of different Hamilton paths in the given graph. So, informally, the complexity class ♯P is the set of the counting problems associated with the decision problems in the class NP. In other words, ♯P consists of counting problems, for which membership in the set of items to be counted can be decided in polynomial time. We should mention here that, even in cases in which a decision problem is polynomial, the corresponding counting problem may be hard to solve (one such example is the *Matching* problem).

To give the reader a more formal description we can say that ♯P consists of all functions f such that $f(x)$ is the number of accepting paths of a nondeterministic Turing machine running in polynomial time on input x. More details about this class can be found in [33] or [2]. Strictly speaking, none of the stochastic programming problems we study can belong to this complexity class. We use the term ♯P-hard for an optimization problem in the same way as NP-hard is used for optimization problems whose recognition version is NP-complete.

We are now ready to proceed with our reduction.

Definition 6.1 (Graph Reliability)**.** Given a directed graph $G = (V, E)$ with m edges ($|E| = m$) and n vertices, determine the reliability of the graph, defined as the probability that two given vertices u and v are connected, if each edge $e \in E$ fails independently with probability $1/2$.

This is equivalent to the problem of counting the number of subgraphs of G, from among all 2^m possible subgraphs, that contain a path from u to v.

Theorem 6.1. *Two-stage stochastic programming with discrete distributions on the parameters is $\sharp$P-hard.*

Proof. Take any instance of Graph Reliability, i.e. a network $G = (V, A)$ with two fixed vertices $u, v \in V$. We introduce an extra edge from v to u, and for each edge $(i,j) \in A$ we introduce a variable y_{ij}. We then give each edge a random weight $\mathbf{q}_{ij}$, except for the edge (v,u) that gets a deterministic weight of 1. Let the weights be independent and identically distributed (i.i.d.) with distribution $\Pr\{\mathbf{q}_{ij} = -2\} = \Pr\{\mathbf{q}_{ij} = 0\} = 1/2$. The event $\{\mathbf{q}_{ij} = -2\}$ corresponds to a failure of the edge (i,j) in the Graph Reliability instance.

Observe now that if the network has a path from u to v, then there is a path from u to v consisting of edges with weight 0 only and vice versa. Denote $A' = A \cup (v,u)$. We now define the following two stage stochastic programming problem:

$$\max\{-cx + Q(x) \,|\, 0 \leq x \leq 1\}$$

with

$$\begin{aligned} Q(x) = \mathbf{E}[\max\{ \sum_{(i,j)\in A} \mathbf{q}_{ij} y_{ij} + y_{vu} \;|\; & \\ & \sum_{i:(i,j)\in A'} y_{ij} - \sum_{k:(j,k)\in A'} y_{jk} = 0 \quad \forall j \in V, \\ & 0 \leq y_{ij} \leq x \quad \forall (i,j) \in A' \}], \end{aligned}$$

where c is a parameter, which will be specified in the following. Looking at the above program more carefully, we can see that for a realization of the random variables, the second stage problem can have value at most x, because $q_{ij} \leq 0$ in any case, and this happens when $y_{vu} = x$. And, moreover, the only case where we can have a positive value is when $y_{vu} > 0$. Observe that the restrictions imposed guarantee that for each vertex, the ingoing "flow" must be equal to the outgoing, if we think of the variables y_{ij} as the flow between two vertices through a specific edge. So, if at least one variable y_{ij} is non-zero, then there must be a closed path of edges with corresponding variables y_{ij}'s non-zero.

Suppose now that for a realization of the failures of the edges there is a path from u to v in the network. As we have already stated, each edge of such a path has cost $q_{ij} = 0$. For such a realization, and having in mind the above observation about the flows, the optimal solution of the second-stage problem is obtained by setting all y_{ij}'s corresponding to edges (i,j) on this path and y_{vu} equal to x, and setting $y_{ij} = 0$ for all edges not on the path. This yields a solution of value x for this realization.

On the other hand, consider a realization in which the graph does not have a path from u to v. This means that on each path between u and v there is at least one edge with weight -2 (and vice versa). Then, the optimal solution has value 0, since y_{vu} cannot be positive because that would imply that a closed path between u and v exists, and this optimal solution of value 0 is obtained by setting all $y_{ij} = 0$ and $y_{vu} = 0$.

Therefore, the network has reliability R if and only if $Q(x) = Rx$. This implies immediately that evaluation of Q in a single point $x > 0$ is $\sharp$P-hard. We continue to prove that the two-stage stage problem is $\sharp$P-hard (it is not excluded that finding the optimal solution to a two-stage problem requires any evaluation of the objective function).

Notice that $Q(x) = Rx$ implies that the objective function value of the two-stage problem is $(R-c)x$. Thus, if $c < R$ then the optimal solution is $x = 1$ with value $(R-c)$, and if $c \geq R$ then the optimal solution is $x = 0$ with value 0. Suppose now that we can solve the above stochastic program. Since R can take only 2^m possible values, then we could perform a bisection search on the values of c and this would allow us to compute the exact value of R by solving only at most m two-stage stochastic programs, since a value 0 would mean that c is on the right half of the interval examined, and a value z would allow us to compute $R = z + c$. Thus, if one could solve the two-stage stochastic programming problem then one could solve the $\sharp$P-hard Graph Reliability problem.

$\square$

The reader may have noticed that the second stage of the two-stage stochastic programming problem used in the proof is not a recourse problem. A similar reduction shows that also the more special class of two-stage stochastic recourse problems are $\sharp$P-hard.

The same reduction can also show that the two-stage stochastic integer programming problem with discretely distributed parameters, i.e. the problem in which second-stage decision variables are restricted to have integer values, is $\sharp$P-hard.

In the two-stage linear programming problem evaluation of Q at any point x is $\sharp$P-easy, since for any realization of the second-stage random parameters a linear program remains to be solved. Given a $\sharp$P-oracle for evaluating Q at any point x, solving two-stage stochastic linear programming problems (with discretely distributed random variables) will require a polynomial number of consultations of the oracle, since Q is a concave function in x, and maximizing a concave function over a convex set is known to be easy [14]. Thus, two-stage stochastic linear programming is in the class $\mathrm{P}^{\sharp\mathrm{P}}$, which is essentially equivalent to $\sharp$P ([33]).

Given a $\sharp$P-oracle for evaluating Q at any point x, a two-stage stochastic integer programming problem lies in NP. In this case the expected value function is in general not convex but discontinuous piecewise linear with a finite number of points x that are candidate for optimality (see [40]). Thus, two-stage stochastic integer programming is in the class $\mathrm{NP}^{\sharp\mathrm{P}} = \mathrm{P}^{\sharp\mathrm{P}}$ ([45]).

Regarding two-stage stochastic programming problems with continuously distributed parameters, $\sharp$P-hardness of an evaluation of the expected value function Q can be established under even the mildest conditions on the distributions. For the proof, a reduction from the problem of computing the volume of the *knapsack polytope* is used, $\sharp$P-completeness of which has been proved in [12].

The main result follows:

Theorem 6.2. *Evaluation of $Q(x)$ of a two-stage stochastic programming problem with continuously distributed parameters is $\sharp$P-hard, even if all stochastic parameters have the uniform* $[0, 1]$ *distribution.*

Membership of this problem in $\sharp$P would require additional conditions on the input distributions.

Part II

Multistage Stochastic Optimization

Chapter 7

The k-stage Stochastic Optimization Problem

It is now time to move on and study stochastic problems with more than 2 stages. Extending the paraphrase of 2-stage problems, one would naturally ask "and what about Wednesday?". The reader should already sense that k-stage problems are generally hard problems, although we will see that there has been some progress and there are some positive results.

A motiving example that may convince the reader that k-stage stochastic problems are indeed worth studying is given in [5]. Consider that we want to minimize the expected cost of operating a water reservoir where one can decide, in each time period, the amount of irrigation water to be sold while maintaining the level of the reservoir within a specified range (where penalties are incurred for violating this constraint). The source of uncertainty is, of course, the variability in rainfall, and there is a simulation model that provides a means to sample from the distribution of inputs (of rainfall amounts per time period within the planning horizon). Observe that it is important to model this as a multistage process, rather than as a 2-stage one, since it allows us to capture essential conditional information, such as given a drought over the previous period, the next period is more likely to continue these conditions.

More formally, in the k-stage problem, information concerning the input is revealed gradually in each of the k stages. In the first stage we are given a probability distribution over possible scenarios and we construct an anticipatory part of the solution, x, incurring a cost c(x). Then, at each stage $i > 1$ new information is received and one can buy some extra elements to augment the current solution. As in 2-stage problems, the cost in each stage increases compared to the previous one.

Our goal is to minimize the expected cost incurred in all stages together. We should make some points clear here. In order to better understand what is actually an optimal solution for a k-stage problem, we can view an optimal solution as a process which suggests a certain and deterministically specified first-stage solution, and then, in each consecutive stage, and taking into account the information received up to that stage, suggests a partial solution, so that in the final stage, where all information is revealed, we have a feasible solution for the scenario realized. In each stage, the solution Z_i^* is a function of the signals $s_1, ..., s_i$, that is $Z_i^* = Z_i^*(s_1, ..., s_i)$. So, choosing these partial solutions leads to a minimization of the expected cost in all stages. In the case where we have the total description of the distribution, and all the conditional probabilities that may occur in each stage, our problem becomes a *fully deterministic* one.

A more illustrative way to describe the k-stage problem is by using a tree. Each level i corresponds to the stage i, and the children of each node in level i correspond to the possible realizations of the signal s_{i+1} from the conditional distribution that is uniquely defined when we follow the path from this node to the root. This also makes explicit that in each stage i we actually have to solve a remaining $(k - i + 1)$-stage problem. Observe that an optimal solution to our problem "includes" the optimal solutions of all $(k-j)$-stage subproblems that can occur, that is, at any node of the tree, the optimal solution of the subproblem that this node defines is part of the optimal solution of the initial problem. In any case, the main issue for any algorithm that deals with k-stage problems is how to choose a first-stage vector so as to minimize the expected cost in all stages.

The last thing to comment on before moving on to algorithms for such problems is how a black-box is defined for such problems. As in 2-stage problems, all algorithms presented will be in the black-box model. In the k-stage problem, a black-box is defined as follows: it has $k-2$ input parameters, corresponding to signals $s_2, s_3, ..., s_{k-1}$ (we remind the reader that signal s_1 is a dummy signal and signal s_k is the realization of the actual input), and depending on the input given, it can give a sample of the next-stage signal according to the conditional distribution defined by the input. More specifically, if until stage i we have the realization $(\mathbf{s}_2, ..., \mathbf{s}_i) = (s_2, ..., s_i)$, then giving as input this exact vector, the black-box will return a sample s_{i+1} of the signal $\mathbf{s}_{i+1}$ from the conditional distribution $\pi[\mathbf{s}_2 = s_2, ..., \mathbf{s}_i = s_i]$. We can then emulate the next stages by giving as input consecutively the values s_j of the signals $\mathbf{s}_j$, $j > i$, produced from the black-box , until we reach a sample S of the actual demands.

Chapter 8

The Multistage Boosted Sampling Algorithm

8.1 Introduction

In this chapter we are going to present an extension of the Boosted Sampling framework presented in chapter 3 that will allow us to deal with k-stage stochastic problems. The extension was given by Gupta etc [18] and led to constant-factor approximations for various problems provided that the number of stages is considered fixed and not part of the input.

8.2 Model

We are again going to use the definition of an abstract deterministic combinatorial optimization problem Π, as given in section 1.3. We will work under the assumption that the inflation factors $\sigma_i \geq 1$ in each stage are constants and deterministically known in advance. This restriction can be waived, in a similar way that was done in the original Boosted Sampling algorithm.

At the beginning of the i^{th} stage (where $1 \leq i \leq k-1$), we receive a signal s_i that represents the information gained about future demands that will arise. After this observation s_i, we know that future signals, as well as the set S of eventual demands, will come from a revised distribution conditioned on seeing this signal. After observing the signal s_i, we can purchase some more elements F_i at cost $(\prod_{j=1}^{i} \sigma_j)c(F_i)$. Finally, in the k^{th} stage we observe the realization of the random variable $\mathbf{S} = S$ of demands, and have to buy the final set F_k so that $\bigcup_{i=1}^{k} F_i \in Sols(S)$. Our goal is to minimize the expected cost incurred in all stages together, that is

$$\mathbf{E}\Big[\sum_{i=1}^{k}(\prod_{j\leq i}\sigma_j)c(F_i)\Big].$$

We will work with problems that are sub-additive (definition 3.1). We are also going to need the notion of cost-shares, introduced in section 3.2.2. The reader who has not gone through that section is suggested to do so before continuing here. We are now going to give some extra properties for cost-sharing functions, that we will need.

Definition 8.1 (Cross-monotonicity)**.** A cost-sharing function ξ is cross-monotone if for every pair of clients $S \subseteq T$ and client $j \in S$, we have $\xi(X,T,j) \leq \xi(X,S,j)$.

This property just states that no demand can cause greater cost, when seen in a superset of demands.

We will now present a subtly different notion of strictness than that presented in section 3.2.2. We create a new reduced instance of the problem Π by zeroing out the cost of all elements in F. This instance is denoted by X/F.

Definition 8.2 (c-strictness)**.** Let $S, T \subseteq U$ be sets of clients, and let X be an instance of Π. The cost-sharing function ξ given by an algorithm $\mathcal{A}$ is β-c-strict if

$$\xi(X/\mathcal{A}(X,S),T,T) \leq \beta \cdot \xi(X, S \cup T, T).$$

This notion of strictness tells us that when we have a solution for a client set S, the cost-shares for a client set T that occur when we try to augment our solution to a solution for T are not much costlier than the cost-shares for T when we know from the start the total client set $S \cup T$. It is similar to the notion of β-strictness, but it is a bit more relaxed condition, as no reference is made to the actual cost of the augmenting solution.

We should note here that a competitive and β-strict cost-sharing function is β-c-strict. This is easily derived as follows. Since ξ is β-strict, there is an augmenting solution F_T such that $c(F_T) \leq \beta \cdot \xi(X, S \cup T, T)$. Observe that the augmenting algorithm cannot do better than $OPT(X/\mathcal{A}(X,S),T)$. Thus, $OPT(X/\mathcal{A}(X,S),T) \leq c(F_T) \Rightarrow OPT(X/\mathcal{A}(X,S),T) \leq \beta \cdot \xi(X, S \cup T, T)$. Using now the fact that ξ is competitive, we get $\xi(X/\mathcal{A}(X,S),T,T) \leq \beta \cdot \xi(X, S \cup T, T)$, and, thus, ξ is β-c-strict.

We will also need the following property which goes hand-in-hand with c-strictness.

Definition 8.3 (approximation w.r.t ξ)**.** An algorithm $\mathcal{A}$ with cost-sharing function ξ is an α-approximation algorithm with respect to ξ if

$$c(\mathcal{A}(X,S)) \leq \alpha\xi(X,S,S).$$

Observe that if ξ is competitive, then $\xi(X,S,S) \leq OPT(S)$, and so an α-approximation algorithm w.r.t ξ gives a stronger guarantee, and also is an α-approximation algorithm. Moreover, an α-approximation algorithm w.r.t a β-c-strict ξ, is $(\alpha\beta)$-strict. This is also easily provable. As already mentioned, a simple augmenting algorithm to convert a solution in S to a solution in T is to solve the instance $(X/\mathcal{A}(X,S),T)$. Thus, in this case $c(F_T) = c(\mathcal{A}(X/\mathcal{A}(X,S),T))$. So, $c(F_T) \leq \alpha\xi(X/\mathcal{A}(X,S),T,T)$, and $\xi(X/\mathcal{A}(X,S),T,T) \leq \beta \cdot \xi(X, S \cup T, T)$. Combining these two we get that there exists an augmenting solution F_T such that $c(F_T) \leq \alpha\beta \cdot \xi(X, S \cup T, T)$.

8.3 The `Multi-Boost-and-Sample` Algorithm

We will now proceed to the main result of [18]. The idea is the same as the two-stage algorithm. In each stage i, we sample $\lfloor \sigma_{i+1} \rfloor$ times from the black-box and we augment the existing solution so as to satisfy the union of the new samples. Finally, in the k^{th} stage the real set of demands is revealed and we augment our solution to get a feasible solution for the realized set.

We present here the algorithm `Multi-Boost-and-Sample`(Π, i) to be executed in stage i. Note that the algorithm is to be executed in *every* stage of the problem. Also, observe that in the special case of 2-stage problems, we get back precisely the original Boosted Sampling framework.

Having explained how the black-box operates in a k-stage problem, the reader should be in a position to imagine how a typical sampling procedure for our problem is. Recursion here fits in very well. What we actually have to do in stage i is emulate $\lfloor \sigma_{i+1} \rfloor$ executions in the remaining $(k-i)$-stages. This way, we can obtain a collection of $\lfloor \sigma_{i+1} \rfloor$ sampled sets of clients. The number of calls to the black-box in stage i, as it will become obvious, is $\prod_{j=i+1}^{k} \lfloor \sigma_j \rfloor$. Observe that the total number of samples generated in the k^{th} level of the tree is the number of total

Algorithm 6: `Multi-Boost-and-Sample`(Π, i)

```
1. If $i < k$, observe the signal $s_i$.
   If $i = k$, observe the required set of clients $S$ instead.

2. If $i = k$, let $D_k := S$. Else $i < k$, and then use procedure
   Recur-Sample$(\Pi, i, s_1, ..., s_i)$ to obtain a sample set of clients $D_i$.

3. Let $B_i = \bigcup_{i=1}^{i-1} F_j$ be the elements that were bought in earlier rounds.
   Set the costs of elements $e \in B_i$ to zero. Using algorithm $\mathcal{A}$, find a set of
   elements $F_i \subseteq X \setminus B_i$ to buy so that $(F_i \cup B_i) \in Sols(D_i)$.
```

Algorithm 7: `Recur-Sample`$(\Pi, i, s_1, ..., s_i)$

```
1. If $i = k$, draw one sample set of clients $S_k$ from the conditional
   distribution $[\pi \mid s_1, ..., s_k]$. Return the set $S_k$ .

2. If $i < k$, draw $\lfloor \sigma_{i+1} \rfloor$ samples of the signal $s_{i+1}$ from the conditional
   distribution $[\pi \mid s_1, ..., s_i]$. Let $s^1, ...s^n$ be the sampled signals
   (where $n = \lfloor \sigma_{i+1} \rfloor$).

3. For each sample signal $s^j, j = 1, ..., n$, recursively call
   Recur-Sample$(\Pi, i+1, s_1, ..., s_i, s^j)$ to obtain a sample set of
   clients $S^j$ . Return the set $S_i = \bigcup_{j=1}^{n} S^j$.
```

calls in the black-box. However, the procedure `Recur-Sample` returns unions of samples so we actually get $\lfloor \sigma_{i+1} \rfloor$ samples when we call it in the i^{th} stage.

We now state the main results obtained.

Theorem 8.1. *Given a problem Π, if $\mathcal{A}$ is an α-approximation algorithm w.r.t. a β-c-strict cost-sharing function ξ, and if ξ is cross-monotone, then* `Multi-Boost-and-Sample`(Π) *is an $\alpha \cdot \sum_{i=0}^{k-1} \beta^i$-approximation algorithm for the k-stage stochastic problem $Stoc_k(\Pi)$.*

Note that the approximation guarantee is on expecation. We should mention here that the additional assumption of cross-monotonicity can be removed at the expense of somewhat worse approximation ratio. Although many problems have cross-monotone cost-shares, there are others in which cross-monotone cost-sharing functions with good guarantees are not available (like Vertex Cover; the interested reader can check [22]). In such cases, we provide the following theorem.

Theorem 8.2. *Given a problem Π, if $\mathcal{A}$ is an α-approximation algorithm w.r.t a β_1-c-strict cost-sharing function ξ that is also β_2-strict, then algorithm* `Multi-Boost-and-Sample`(Π) *is an $\alpha \cdot \sum_{i=0}^{k-1} (\beta_1\beta_2)^i$ approximation algorithm for the k-stage stochastic problem $Stoc_k(\Pi)$.*

We are now going to prove theorem 8.1, and then explain how the proof can be modified in order to prove theorem 8.2. As in the proof of the corresponding 2-stage theorem of chapter 3, we will try to separately bound the expected cost of the solution F_i in stage i. However, a similar approach to the one used in the 2-stage theorem does not succeed here, since we have to move between the cost-shares and the cost of the solutions F_i, which causes us to lose factors of $\approx \alpha$ at each step (observe that the α-approximation w.r.t a β-c-strict function implies, as already mentioned, a $(\alpha\beta)$-strict function, and thus we would have a loss factor $\alpha\beta$ in each stage). Instead, we bound all costs incurred in terms of the ξ's, which can be done since the approximation algorithm is w.r.t. a cost-sharing function. We first argue that the expected sum of cost-shares paid in the first stage is no more than the optimum total expected cost Z^*, and then bound the sum of cost-shares in each consecutive stage in terms of the expected cost shares from the previous stage (with a loss of a factor of β at each stage). Finally, we bound the actual cost of the partial solution constructed at stage i by α times the expected cost-shares for that stage, which gives us the geometric sum claimed in the theorem.

The proof will be presented in steps, as two lemmas will make it much easier. The first concerns the cost of the first-stage and the other the cost of stage i. Let F^* be an optimal solution to the given instance of $Stoc_k(\mathit{\Pi})$. We denote by F_i^* the partial solution built in stage i; recall that $F_i^* = F_i^*(s_1, s_2, ..., s_i)$ is a function of the set of all possible i-tuples of signals that could be observed before stage i. The expected cost of this solution can be expressed as

$$Z^* = \sigma_1 \mathbf{E}[c(F_1^*(\mathbf{s}_1))] + \sigma_1\sigma_2 \mathbf{E}[c(F_2^*(\mathbf{s}_1, \mathbf{s}_2))] + ... + \sigma_1 ... \sigma_k \mathbf{E}[c(F_k^*(\mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_k))].$$

Lemma 8.1. *The expected cost-share* $\mathbf{E}[\xi(X, D_1, D_1)]$ *is at most the total optimum cost* Z^*.

Proof. Let D_1 be the sample set of clients returned by `Recur-Sample`$(\mathit{\Pi}, 1, s_1)$. We will prove that there is a solution $\widehat{F}(D_1) \in Sols(D_1)$ such that $\mathbf{E}[c(\widehat{F}(D_1))] \leq Z^*$ (where the expectation is over the execution of the procedure `Recur-Sample`). Consider now the tree of recursive calls of the procedure `Recur-Sample`. Each call `Recur-Sample`$(\mathit{\Pi}, i, s_1, ..., s_i)$ corresponds to a node in level i of the tree. For each such node we add the set of elements $F_i^*(s_1, ..., s_i)$ to our solution. Since the optimal solution is a feasible solution for any particular realization of the signals (in fact, it is a function of these signals, as any other non-trivial solution), it is obvious that the solution reached, $\widehat{F}(D_1)$, is a feasible solution for the set D_1. Now, since ξ is competitive, we have that $\xi(X, D_1, D_1) \leq OPT(D_1) \leq c(\widehat{F}(D_1))$. Thus, the expected cost is

$$\mathbf{E}[\xi(X, D_1, D_1)] \leq \mathbf{E}[c(\widehat{F}(D_1))] \leq \mathbf{E}\big[\sum_{i=1}^{k} \big(\prod_{j \leq i} \sigma_j\big) c(F_i^*)\big] = Z^*.$$

□

Lemma 8.2. *Let* $\widehat{F} = F_1 \cup ... \cup F_{i-1}$ *be the solution constructed in a particular execution of the first* $i-1$ *stages, and let* s_i *and* s_{i+1} *be the signals observed in stages* i *and* $i+1$ *respectively. Let* D_i *and* D_{i+1} *be the random variables denoting the samples returned by the procedure* `Recur-Sample` *in stages* i *and* $i+1$, *and let* F_i *be the (random) solution constructed by* $\mathcal{A}$ *for the set of clients* D_i. *Then,*

$$\mathbf{E}[\xi(X/(\widehat{F} \cup F_i), D_{i+1}, D_{i+1})] \leq \frac{\beta}{\lfloor \sigma_{i+1} \rfloor} \cdot \mathbf{E}[\xi(X/F, D_i, D_i)].$$

Proof. Recall that the sampling procedure `Recur-Sample`$(\mathit{\Pi}, i, s_1, ..., s_i)$ gets $n = \lfloor \sigma_{i+1} \rfloor$ independent samples $s_1, s_2, ..., s_n$ of the signal s_{i+1} from the distribution π conditioned on $s_1, ..., s_i$, and then for each sampled signal calls itself recursively to obtain the n sets $S^1, ..., S^n$. Note that the set $D_i = \bigcup_{j=1}^n S_j$ is simply the union of these n sets. On the other hand, the set D_{i+1} is obtained by observing the signal s_{i+1} (which is assumed to come from the same distribution $[\pi \mid s_1, ..., s_i]$), and then calling `Recur-Sample` with the observed value of s_{i+1}.

In a similar way as done in the proof of the 2-stage theorem, we will consider an alternate, probabilistically equivalent, view of this process. We first take $n+1$ samples $s^1, ..., s^{n+1}$ of the signal $\mathbf{s}_{i+1}$ from the distribution $[\pi \mid s_1, ..., s_i]$. Calling the procedure `Recur-Sample`$(\mathit{\Pi}, i+1, s_1, ..., s_i, s^j)$ for each s^j we obtain the sets $S^1, ..., S^{n+1}$. We now pick an index j uniformly at random from the set $\{1, ..., n+1\}$. Let $D_{i+1} = S^j$, and let D_i be the union of the remaining

n sets. This process of randomly constructing the pair of sets (D_i, D_{i+1}) is equivalent to the original process, as a realization of a signal is equivalent to drawing a sample from the black-box. Note that $D_i \cup D_{i+1} = \bigcup_{l=1}^{n+1} S^l$.

For simplicity of notation, let us denote $X/\widehat{F}$ by $\widehat{X}$. By the definition of β-c-strictness, we first get

$$\xi(\widehat{X}/F_i, D_{i+1}, D_{i+1}) \leq \beta \cdot \xi(\widehat{X}, D_i \cup D_{i+1}, D_{i+1}).$$

Taking into account that D_{i+1} is equivalent to S^j sampled uniformly from $n+1$ alternates in the equivalent process above and that D_i is the union of the remaining n sets, we have

$$\mathbf{E}[\xi(\widehat{X}, D_i \cup D_{i+1}, S^j)] \leq \mathbf{E}\bigg[\frac{1}{n} \cdot \xi(\widehat{X}, D_i \cup D_{i+1}, D_i)\bigg].$$

Now we use cross-monotonicity of the cost shares and finally get

$$\mathbf{E}[\xi(\widehat{X}, D_i \cup D_{i+1}, D_i)] \leq \mathbf{E}[\xi(\widehat{X}, D_i, D_i)].$$

By consecutively using the above inequalities we get our result.

□

Proof of theorem 8.1. Recall that the expected cost of the solution given by the algorithm `Multi-Boost-and-Sample` is

$$\mathbf{E}[Z] = \mathbf{E}\bigg[\sum_{i=1}^{k} (\prod_{j=1}^{i} \sigma_j) c(F_i)\bigg].$$

Using cross-monotonicity and the algorithm $\mathcal{A}$ in the instance $(X/B_i, D_i)$, where $\mathcal{A}$ is an α-approximation algorithm w.r.t the β-c-strict cost-shares ξ, we have

$$\mathbf{E}[Z] \leq \alpha \mathbf{E}\bigg[\sum_{i=1}^{k} (\prod_{j=1}^{i} \sigma_j) \xi(X/B_i, D_i, D_i)\bigg].$$

We can now use lemma 8.2 inductively on $\xi(X/B_i, D_i, D_i)$ to get that $\xi(X/B_i, D_i, D_i) \leq \frac{\beta^{i-1}}{\prod_{j=1}^{i} \lfloor \sigma_j \rfloor} \xi(X, D_1, D_1)$. Using this inequality in the bound for $\mathbf{E}[Z]$ we get

$$\mathbf{E}[Z] \leq \alpha \mathbf{E}\bigg[\sum_{i=1}^{k} \beta^{i-1} \xi(X, D_1, D_1)\bigg].$$

Using lemma 8.1 in the above inequality gives our result.

□

In order now to prove theorem 8.2, which requires no cross-monotone cost-shares, we need to prove a variation of lemma 8.2. We state the lemma below. The proof is omitted, as it is very similar to the one of lemma 8.2 and we do not believe it gives any more insight of cost-shares.

Lemma 8.3. *Let $\widehat{F} = F_1 \cup ... \cup F_{i-1}$ be the solution constructed in a particular execution of the first $i-1$ stages, and let s_i be the signal observed in stage i. Let D_i and D_{i+1} be the random variables denoting the samples returned by the procedure* `Recur-Sample` *in stages i and $i+1$, and let F_i be the (random) solution constructed by $\mathcal{A}$ for the set of clients D_i. Then,*

$$\mathbf{E}[\xi(X/(\widehat{F} \cup F_i), D_{i+1}, D_{i+1})] \leq \frac{\beta_1 \beta_2}{\lfloor \sigma_{i+1} \rfloor} \cdot \mathbf{E}[\xi(X/\widehat{F}, D_i, D_i)].$$

8.4 Applications

In this section, we will see some applications of the framework presented in this chapter. We will not go into any technical details regarding individual problems.

8.4.1 The Rooted Steiner Tree Problem

Theorem 8.3. *There is a 2-approximation algorithm for the Minimum Steiner Tree problem w.r.t. a 1-c-strict cost-sharing function ξ. Furthermore, this ξ is also cross-monotone.*

The cost-sharing function ξ for Steiner Tree given by Jain and Vazirani [23] is cross-monotone, and the minimum spanning tree heuristic is 2-strict w.r.t it. Moreover, it can be shown to be 1-c-strict. So, we get the following result:

Theorem 8.4. *There is a $2k$-aproximation algorithm for the k-stage Stochastic Rooted Steiner Tree Problem.*

8.4.2 The Metric Uncapacitated Facility Location Problem

Theorem 8.5. *There is a 3-approximation algorithm for the Metric Uncapacitated Facility Location problem w.r.t a 2-c-strict cost-sharing function ξ. Furthermore, this ξ is also cross-monotone.*

The cost-sharing function used is a slight variant of the cost-sharing function defined by Pál and Tardos [34]. We get the following result:

Theorem 8.6. *There is a $3(2^k - 1)$-approximation algorithm for the k-stage Stochastic MUFLP.*

8.4.3 Vertex Cover

Theorem 8.7. *There is a 2-approximation algorithm for the Vertex Cover problem w.r.t. a 2-strict and competitive (and hence 2-c-strict) cost-sharing function ξ.*

The cost-sharing function used is given in [17]. Note that this function is not cross-monotone, and thus we use the weaker version of our theorem. We get the following result:

Theorem 8.8. *There is a $\frac{2}{3}(4^k - 1)$-approximation algorithm for the k-stage Stochastic Vertex Cover.*

Chapter 9

The SAA Method for Multistage Problems

9.1 Introduction

In this chapter, we are going to see how the SAA method can be applied in multistage stochastic optimization problems. We are going to go through the work of Swamy and Shmoys [43], who prove that polynomial bounds on the sample size are adequate in order to yield a fully polynomial approximation scheme for a broad class of multistage stochastic linear programs with any *constant* (i.e. not part of the input) number of stages. This, combined with rounding techniques, can give approximation algorithms for various combinatorial problems. We will see the results obtained from both the deterministic rounding Swamy and Shmoys suggest in [38, 43] and the randomized rounding Srinivasan uses in [39] for covering and facility location problems.

Many ideas presented here are based on techniques developed in chapter 4, and although we will try to make this chapter as self-contained as possible, we will make some references to that chapter, so the reader who has skipped it is suggested to take a look at it before proceeding here.

9.2 An outline of our approach

We once again consider black-box problems. We describe briefly the SAA approach for multistage problems: sample some N times from the distribution on scenarios, estimate the actual distribution by the distribution induced by the samples, and solve the multistage problem specified by the approximate distribution. For 2-stage programs, as already presented, we just estimate the probability of scenario A by its frequency in the sampled set; for k-stage programs we construct an approximate k-level distribution tree by sampling repeatedly for each level: we sample $\mathcal{T}_2$ times to obtain some stage 2 outcomes, for each such outcome we sample $\mathcal{T}_3$ times from the conditional distribution given that outcome to generate some stage 3 outcomes and so on, and for each sampled outcome we estimate its conditional probability of occurrence given the previous-stage outcome by its frequency in the sampled set. The multistage problem specified by the approximate distribution is called the sample average problem, and its objective function is called the sample average function. If the total number of samples $\mathcal{N}$ is polynomially bounded, then since the approximate distribution has support of size at most $\mathcal{N}$, the sample average problem can be solved efficiently by solving a polynomial size linear program.

Our goal is to prove that a polynomial sample size suffices so that with high probability, every optimal solution to the SAA problem is near-optimal to the true problem. We are going to deal with linear programs, which can be viewed as relaxations of integer programs corresponding to combinatorial problems. So, reaching such a result would mean that an optimal solution to the SAA problem, which is a polynomial size problem and thus easily solvable, would be a "good" solution to the true problem.

An initial approach towards this would be to try and prove that the values of the SAA function and the true function are close to each other. Observe that this would imply that we can approximate the value of the true function at any point. This however immediately runs into problems since the variance in the scenario costs could be exponentially large, so that one cannot hope to estimate the true function value to within a reasonable accuracy with a polynomial number of samples. This is due to some extremely low-probability high cost outcomes which contribute significantly towards the cost in the true problem, but will almost never be sampled with only a polynomial number of samples, and so they contribute nothing to the SAA function. We remind the reader that such an observation was also made in chapter 5, where we used a somewhat weaker notion of high cost samples. The key fact here is that such rare outcomes do not significantly influence the first-stage decision, since it is reasonable to defer decisions for such outcomes till later.

Taking into account the results obtained in chapter 4, we could try convex optimization techniques in order to show the near-equivalence of the two problems. The minimizer of a convex function is determined by its "slope" (i.e. its gradient or subgradient). In our case, and as already suggested in chapter 4, we can use subgradients as a measure of the "slope" of our functions and show that they are close to each other. We can then argue that this is sufficient to prove the aforementioned near-equivalence of the two (true and SAA) minimization problems. More specifically, we identify a notion of closeness between any two functions based on their subgradients so that if two functions are close under this criterion, then minimizing one is approximately equivalent to minimizing the other. Next, we show that the objective functions of the original multistage problem, and the sample average problem with polynomially bounded sample size, satisfy this "closeness-in-subgradients" property with high probability, and thus we obtain the desired result.

To get some intuition about this "closeness-in-subgradients" property, one can think of the ellipsoid algorithm of Shmoys and Swamy presented in chapter 4. Remember that our goal there was to solve the problem: $\min_{x\in\mathcal{P}} h(x)$, where h is a convex function defined in a (convex) polytope $\mathcal{P}$. As we were not able to calculate any value of h, the algorithm used only information about the subgradient of h in order to make progress. Thus, two functions that satisfy this "closeness-in-subgradients" property in a (convex) polytope $\mathcal{P}$, would make the algorithm run identically on the corresponding minimization problems. And although this only proves that one (and not every) optimal solution is near optimal to the other problem, we will formalize this "closeness-in-subgradients" property and prove that it is a sufficient condition to prove the near-equivalence of the two problems, even when satisfied only in a (dense) finite grid of polytope $\mathcal{P}$.

We now turn our attention to choosing an appropriate subgradient that will be convenient in proving the "closeness" property. We remind the reader here of the "nice" subgradient presented in lemma 4.8, and the fact that we could component-wise approximate it through a sampling process. We will try to expand this idea. The crucial point which makes things more difficult in multistage problems compared to 2-stage problems is that the above subgradient is constructed from the dual of the recourse problem. In 2-stage problems, the true problem and the SAA problem have the same recourse problem, which is a standard LP, and so the property is not hard to prove. However, for $k \geq 3$ stages, the recourse problem is a $(k-1)$-stochastic problem, and so the true recourse and the SAA recourse are different because of the difference of their distributions (the first having the true distribution, while the second having the sampled one produced by the SAA method).

We will now describe how we can get over this difficulty by focusing on a 3-stage problem. The recourse problem in such a case is a 2-stage stochastic problem. What we need to show is that by solving the dual of the SAA recourse we have a good solution for the true dual. In other words, we need to show an SAA theorem for the 2-stage recourse problems, which leads us to try and prove the closeness property for the max-subgradients of the two problems. However, we cannot do so, as the duals have generally different feasible region (this is due to the polynomial approximation of the original distribution). So, what we actually need is to formulate the recourse problems differently. By using a Lagrangian dual of the recourse, we formulate the dual as a concave maximization problem with a 2-stage primal LP embedded in it. Now, a max-subgradient of the dual can be calculated from this embedded 2-stage LP. Thus, the only thing left to prove is an SAA theorem for these embedded 2-stage primal programs, which can be done.

Using this idea inductively, one can generalize in k-stage problems, where k is any fixed number. We will see all these in more detail in the following section. But before proceeding to our analysis, we should make one last comment. The main idea of our approach is the "closeness-in-subgradients" property that the true and the SAA function satisfy. One can look at it in a different way. Since we want to actually solve a simpler problem, we need to prove that the true problem is near-equivalent to a simpler one. The properties of subgradients can guide us to look for simpler objective functions which have subgradients that are approximate subgradients for the true function. Now, by observing that approximate subgradients of the true function can be calculated through sampling and averaging, we are led to try the SAA function. And we see that it really fits in very well.

9.3 Analysis

We now proceed to see all the ideas mentioned above in more detail. We at first give some definitions that we are going to need. We will need a slightly different, weaker definition of approximate subgradients than the one given in 4.2. Note that the algorithm presented in chapter 4 can be implemented by using this notion of approximate subgradient. Let $g : \mathbb{R}^m \to \mathbb{R}$.

Definition 9.1. We say that $\widehat{d}$ is an $(\omega, \Delta, \mathcal{D})$-subgradient of g at the point $u \in \mathcal{D}$ if for every $v \in \mathcal{D}$, we have that $g(v) - g(u) \geq \widehat{d} \cdot (v - u) - \omega g(u) - \omega g(v) - \Delta$.

We will consider convex minimization problems $\min_{x \in \mathcal{P}} g(x)$ where $\mathcal{P} \subseteq \mathbb{R}^m_{\geq 0}$ is a polytope and $g(.)$ is convex. We remind the reader of the definition of Lipschitz constant 4.3 and a relevant lemma 4.9.

We will also encounter concave maximization problems $\max_{x \in \mathcal{P}} g(x)$, where $g(.)$ is concave. Analogous to the definition of a subgradient, we define a max-subgradient and an approximate version of a max-subgradient.

Definition 9.2. We say that d is a max-subgradient of a function $g : \mathbb{R}^m \to \mathbb{R}$ at $u \in \mathbb{R}^m$ if for every point $v \in \mathbb{R}^m$, we have $g(v) - g(u) \leq d \cdot (v - u)$. We say that $\widehat{d}$ is an $(\omega, \Delta, \mathcal{D})$-max-subgradient of g(.) at $u \in \mathcal{D}$ if for every $v \in \mathcal{D}$ we have $g(v) - g(u) \leq d \cdot (v - u) + \omega g(u) + \Delta$.

When $\mathcal{D}$ is clear from the context, we abbreviate $(\omega, \Delta, \mathcal{D})$-subgradient and $(\omega, \Delta, \mathcal{D})$-max-subgradient to (ω, Δ)-subgradient and (ω, Δ)-max-subgradient respectively. If $\Delta = 0$, we will use $(\omega, \mathcal{D})$-subgradient and $(\omega, \mathcal{D})$-max-subgradient, instead of $(\omega, \Delta, \mathcal{D})$-subgradient and $(\omega, \Delta, \mathcal{D})$-max-subgradient respectively. We will frequently use $(\omega, \Delta, \mathcal{P})$-subgradients which we abbreviate and denote as (ω, Δ)-subgradients from now on.

We will also need the following sampling lemma which is proved using simple Chernoff bounds (we remind the reader of the similar lemma 4.10).

Lemma 9.1. *Let X_i, $i = 1, ..., N = \frac{4(1+k)^2}{c^2} \ln\left(\frac{2}{\delta}\right)$ be iid random variables where each $X_i \in [-a, b], a, b > 0, k = \max(1, a/b)$, and $c \in [0, 1]$. Let $X = (\sum_i X_i)/N$ and $\mu = \mathbf{E}[X] = \mathbf{E}[X_i]$. Then $\Pr\big[X \in [\mu - cb, \mu + cb]\big] \geq 1 - \delta$.*

Proof. Let $Y_i = X_i + a \in [0, a+b]$ and $Y = \sum_i Y_i$. Let $\mu' = \mathbf{E}[Y_i] = \mu + a$. We want to calculate the probability of the events $[X > \mu + cb]$ and $[X < \mu - cb]$. We have $\mathbf{E}[Y] = N(\mu + a)$, so

$$\begin{aligned}
X &> \mu + cb \Leftrightarrow \\
\sum_i X_i &> N\mu + Ncb \Leftrightarrow \\
\sum_i Y_i &> N\mu + Ncb + Na \Leftrightarrow \\
Y &> \mathbf{E}[Y] + Ncb \Leftrightarrow \\
Y &> \mathbf{E}[Y]\Big(1 + \frac{Ncb}{N(\mu + a)}\Big) \Leftrightarrow \\
Y &> \mathbf{E}[Y]\Big(1 + \frac{cb}{\mu'}\Big)
\end{aligned}$$

Let $v = cb/\mu'$. Thus we have that $\Pr[X > \mu + cb] = \Pr[Y > \mathbf{E}[Y](1 + v)]$. In a similar way we get $\Pr[X < \mu - cb] = \Pr[Y < \mathbf{E}[Y](1 - v)]$. Note that $\mu' \leq a + b$. Since the variables Y_i are independent we can use Chernoff bounds. We have that $\Pr[X < \mu - cb] = \Pr[Y < \mathbf{E}[Y](1 - v)]$ is at most $e^{-\frac{v^2 N\mu'}{2(a+b)}} = e^{-\frac{(cb)^2 N}{2\mu'(a+b)}} \leq \frac{\delta}{2}$. To bound $\Pr[Y > \mathbf{E}[Y](1 + v)]$ we consider two cases. If $v > 2e - 1$, then this quantity is at most $2^{-\frac{(1+v)N\mu'}{a+b}}$ which is bounded by $2^{-\frac{vN\mu'}{a+b}} \leq \frac{\delta}{2}$. If $v \leq 2e - 1$, then the probability is at most $e^{-\frac{v^2 N\mu'}{4(a+b)}} = e^{-\frac{(cb)^2 N}{4\mu'(a+b)}} \leq \frac{\delta}{2}$. So, using the union bound, we get that $\Pr\big[X \notin [\mu - cb, \mu + cb]\big] \leq \delta$.

□

We are now ready to prove that the "closeness-in-subgradients" property is sufficient for our goal.

9.3.1 Sufficiency of closeness in subgradients

Let $g : \mathbb{R}^m \to \mathbb{R}$ and $\widehat{g} : \mathbb{R}^m \to \mathbb{R}$ be two functions with Lipschitz constant (at most) K. Let $\mathcal{P} \subseteq \mathbb{R}^m_{\geq 0}$ be the bounded feasible region, R be a radius such that $\mathcal{P}$ is contained in the ball $B(\mathbf{0}, R) = \{x : x \leq R\}$, and V be a radius such that $\mathcal{P}$ contains a ball of radius V (where $V \leq 1$ without loss of generality). Let z be a point in $\mathcal{P}$. Let $\epsilon, \gamma > 0$ be two parameters with $\gamma \leq 1$. Set $N = \log_2(\frac{2KR}{\epsilon})$ and $\omega = \frac{\gamma}{8N}$.

As already mentioned, we will prove the "closeness-in-subgradients" property in a finite grid. For this purpose, we will define two grids of the polytope $\mathcal{P}$. Let $G' = \{x \in \mathcal{P} : x_i - z_i = n_i \cdot \left(\frac{\epsilon V}{8KNR\sqrt{m}}\right),\ n_i \in \mathbb{Z}$ for all $i = 1, ..., m\}$. This is an orthogonal grid with cell size $\frac{\epsilon V}{8KNR\sqrt{m}}$ in each dimension. We now extend this grid in the following way. Set $G = G' \cup \{x + t(y - x), y + t(x - y) : x, y \in G',\ t = 2^{-i},\ i = 1, ..., N\}$.

We call G' and G the $\frac{\epsilon V}{8KNR\sqrt{m}}$-grid and the *extended* $\frac{\epsilon V}{8KNR\sqrt{m}}$-grid of the polytope $\mathcal{P}$, respectively, keeping in mind that these two grids are always "around" a fixed point in $\mathcal{P}$.

We will now prove two useful properties of our grid. Let vol_m denote the volume of the unit ball in m dimensions.

Lemma 9.2. *Let G' be an ϵ-grid of $\mathcal{P}$ around some $z \in \mathcal{P}$, and G be the corresponding extended grid. Then, the volume of a grid cell of G' is at least $\left(\frac{\epsilon}{2}\right)^m vol_m$. Hence, $|G'| \leq \left(\frac{2R}{\epsilon}\right)^m$ and $|G| \leq N|G'|^2$, where $|G'|$ and $|G|$ denote the number of cells that each grid contains.*

Proof. Each grid cell of G' has size ϵ in each dimension, and thus it contains a ball of radius $\epsilon/2$. So, the volume of the cell is at least $\left(\frac{\epsilon}{2}\right)^m vol_m$ (see also lemma 4.1). Since the cells of G' are pairwise disjoint and taking into account that the polytope $\mathcal{P}$ is contained in the ball $B(\mathbf{0}, R)$ which has volume $R^m vol_m$, we get that $|G'| \leq \frac{R^m vol_m}{(\epsilon/2)^m vol_m} = \left(\frac{2R}{\epsilon}\right)^m$. Now, by the definition of G we get $|G| \leq |G'| + 2N\binom{|G'|}{2} \leq N|G'|^2$.

□

The following lemma now shows that the grid we have chosen is sufficiently dense.

Lemma 9.3. *Let G' be the $\frac{\epsilon V}{8KNR\sqrt{m}}$-grid of $\mathcal{P}$ around z. Then for every $x \in \mathcal{P}$, there exists $x' \in G'$ such that $\|x - x'\| \leq \frac{\epsilon}{KN}$.*

We omit the formal proof, as the techniques used are very similar to the ones used in lemma 4.4. We again use a scaling argument and shrink our initial polytope around x. We then use the center of the ball $B(\mathbf{0}, R)$ that contains polytope $\mathcal{P}$ and which has the property that each other point of $\mathcal{P}$ is at distance at most R from it, and by taking advantage of the scaling in distances in the shrinked polytope we prove that there exists a point in our grid that is near to x.

We are ready to state the "closeness-in-subgradients" property in a more formal way. Fix $\Delta > 0$. We first consider minimization problems. We say that g and $\widehat{g}$ satisfy property (A) if

$$\forall x \in G,\ \exists \widehat{d}_x \in \mathbb{R}^m : \widehat{d}_x \text{ is a subgradient of } \widehat{g}(.) \text{ and an } (\omega, \Delta)\text{-subgradient of } g(.) \text{ at } x. \quad (A)$$

We should mention here that we could replace the grid G' defined above by any grid sufficiently dense so that is satisfies lemma 9.3, and use the corresponding extended grid. The only restriction we will need for multistage programs is that $\ln|G'|$ (and hence $\ln|G|$) should be polynomially bounded. The following lemma now shows that the "closeness-in-subgradients" property is sufficient to prove that an every optimal solution for function $\widehat{g}$ is near-optimal for function g.

Lemma 9.4. *Suppose g and $\widehat{g}$ are functions that satisfy property (A). Let $x^*, \widehat{x} \in \mathcal{P}$ be points that respectively minimize $g(.)$ and $\widehat{g}(.)$ over $\mathcal{P}$, and suppose $g(x^*) \geq 0$. Then, $g(\widehat{x}) \leq (1 + \gamma)g(x^*) + 6\epsilon + 2N\Delta$.*

Proof. Let $\tilde{x}$ be the point in G' closest to x^*, so $\|\tilde{x} - x^*\| \leq \frac{\epsilon}{KN}$ (by lemma 9.3). Since g has Lipschitz constant K we get that $g(\tilde{x}) \leq g(x^*) + \epsilon$. We will consider two cases. Consider first the case when $\widehat{x} \in G'$. We will argue that there is a point x near $\widehat{x}$ such that $g(x)$ is close to $g(x^*)$, and from this it will follow that $g(\widehat{x})$ is close to $g(x^*)$.

Let $y = \widehat{x}\left(1 - \frac{1}{2N}\right) + \left(\frac{1}{2N}\right)\tilde{x} \in G$, since both $\widehat{x}, \tilde{x} \in G'$. From property (A) we get a subgradient $\widehat{d}_y$ at point y for function $\widehat{g}$. From the definition of subgradients, we have $\widehat{g}(\widehat{x}) - \widehat{g}(y) \geq \widehat{d}_y \cdot (\widehat{x} - y)$. Since $\widehat{x}$ is an optimal solution, it must be that $\widehat{d}_y \cdot (\widehat{x} - y) \leq 0$. Observing the "direction" of vectors $\widehat{x} - y$ and $\tilde{x} - y$, or by simply substituting $\widehat{x}$ in the above inequality with the equation that defines y, we get $\widehat{d}_y \cdot (\tilde{x} - y) \geq 0$.

By the definition of an (ω, Δ)-subgradient we have

$$\begin{aligned}
g(y) &\leq \frac{(1+\omega)g(\tilde{x}) + \Delta}{1 - \omega} \\
&\leq (1 + 4\omega)(g(\tilde{x} + \Delta) \qquad \left(\text{since } \omega = \frac{\gamma}{8N} \leq \frac{1}{4}\right) \\
&\leq (1 + \gamma)(g(x^*) + \epsilon + \Delta) \\
&\leq (1 + \gamma)g(x^*) + 2\epsilon + 2\Delta \qquad (\gamma \leq 1)
\end{aligned}$$

Also, $\|\widehat{x} - y\| = \frac{\|\widehat{x}-\tilde{x}\|}{2^N} \leq \frac{\epsilon}{K}$ since $\|\widehat{x} - \tilde{x}\| \leq 2R$. So, due to the Lipschitz constant we get $g(\widehat{x}) \leq g(y) + \epsilon \leq (1+\gamma)g(x^*) + 3\epsilon + 2\Delta$.

Consider now the case when $\widehat{x} \notin G'$. Let $\bar{x}$ be the point in G' closest to $\widehat{x}$, so that $\|\bar{x} - \widehat{x}\| \leq \frac{\epsilon}{KN}$, and $\widehat{g}(\bar{x}) \leq \widehat{g}(\widehat{x}) + \frac{\epsilon}{N}$. For any $y \in G$, if we consider $\widehat{d}_y$ given by property (A), it need not be that $\widehat{d}_y \cdot (\bar{x} - y) \leq 0$, so we have to argue a little differently. Observe however that, since $\widehat{g}(\bar{x}) \leq \widehat{g}(\widehat{x}) + \frac{\epsilon}{N}$, we have that for any $y \in G$, $\widehat{d}_y \cdot (\bar{x} - y) \leq \frac{\epsilon}{N}$, otherwise $\widehat{x}$ would not be an optimal solution. So, we consider the set of points y_i, with $y_0 = \tilde{x}$ and $y_i = (\bar{x} + y_{i-1})/2$, for $i = 1, ..., N$. Note that each $y_i \in G$. Thus, we have $\widehat{d}_{y_i} \cdot (y_{i-1} - y_i) = -\widehat{d}_{y_i} \cdot (\bar{x} - y_i) \geq -\frac{\epsilon}{N}$. Since $\widehat{d}_{y_i}$ is an (ω, Δ)-subgradient of $g(.)$ at y_i, by simple calculus we get $g(y_i) \leq (1+4\omega)\big(g(y_{i-1}) + \frac{\epsilon}{N} + \Delta\big)$. This implies that $g(y_N) \leq (1+4\omega)^N (g(\tilde{x}) + \epsilon + N\Delta) \leq (1+\gamma)g(x^*) + 4\epsilon + 2N\Delta$. So $g(\widehat{x}) \leq g(y_N) + 2\epsilon \leq (1+\gamma)g(x^*) + 6\epsilon + 2N\Delta$.

□

As a corollary of the above lemma, we get the following result concerning approximate and not exact minimizers of function $\widehat{g}$.

Corollary 9.1. *Suppose g and $\widehat{g}$ are functions that satisfy property (A). Let $x^*, \widehat{x} \in \mathcal{P}$ be points that respectively minimize $g(.)$ and $\widehat{g}(.)$ over $\mathcal{P}$, and suppose $g(x^*) \geq 0$. Also, let $x' \in \mathcal{P}$ be such that $\widehat{g}(x') \leq \widehat{g}(\widehat{x}) + p$. Then, $g(x') \leq (1+\gamma)g(x^*) + 6\epsilon + 2N\Delta + 2Np$.*

The proof is very similar to the proof of lemma 9.4 and it is omitted. Also, with similar reasoning we can get analogous results for maximization problems. We state the corresponding property for such problems. We say that g and $\widehat{g}$ satisfy property (B) if

$$\forall x \in G,\ \exists \widehat{d}_x \in \mathbb{R}^m : \widehat{d}_x \text{ is a max-subgradient of } \widehat{g}(.) \text{ and an } (\omega,\Delta)\text{-max-subgradient of } g(.) \text{ at } x. \qquad (B)$$

The analogous result is now this.

Lemma 9.5. *Suppose g and $\widehat{g}$ are functions that satisfy property (B). Let $x^*, \widehat{x} \in \mathcal{P}$ be points that respectively maximize $g(.)$ and $\widehat{g}(.)$ over $\mathcal{P}$, and suppose $g(x^*) \geq 0$. Then, $g(\widehat{x}) \leq (1-\gamma)g(x^*) - 4\epsilon - N\Delta$.*

We now focus on finding a "nice" subgradient that will let us prove the closeness property. We will begin by studying 2-stage problems. The next section is dedicated to proving this for 2-stage problems, and it is the only section of this part that could actually be moved in the first part of this thesis, where we dealt with 2-stage problems exclusively. However, we have decided to put the analysis here, as we think it will help the reader to better understand the difficulties that occur for problems of 3 or more stages by making the comparison between such problems. In any case, the section that follows could be put side by side with chapter 5.

9.3.2 The SAA bound for 2-stage programs

We will now consider the class of problems that we have studied in section 4.3. We restate the problem.

(2Gen-P):

$$\begin{aligned}
\text{min:}\quad & h(x) = w^I \cdot x + \sum_{A \in \mathcal{A}} p_A f_A(x) \quad \text{subject to } x \in \mathcal{P} \subseteq \mathbb{R}^m_{\geq 0}, \\
\text{where}\quad & f_A(x) = \text{min:} \quad w^A \cdot r_A + q^A \cdot s_A \\
& \qquad\qquad \text{s.t:} \quad D^A s_A + T^A r_A \geq j^A - T^A x \\
& \qquad\qquad\qquad\quad r_A \in \mathbb{R}^m_{\geq 0},\ s_A \in \mathbb{R}^n_{\geq 0}.
\end{aligned}$$

We remind the reader of the assumptions made, (a) $T^A \geq \mathbf{0}$ for every scenario A, and (b) for every $x \in \mathcal{P}, \sum_{A\in\mathcal{A}} p_A f_A(x) \geq 0$ and the primal and dual problems corresponding to $f_A(x)$ are feasible for every scenario A. It is assumed that $\mathcal{P} \subseteq B(\mathbf{0}, R)$ and $\mathcal{P}$ contains a ball of radius V ($V \leq 1$) where $\ln(R/V)$ is polynomially bounded. To prevent an exponential blowup in the input, we consider an oracle model where an oracle supplied with scenario A reveals the scenario-dependent data $(w^A, q^A, j^A, D^A, T^A)$. We also define $\lambda = \max(1, \max_{A\in\mathcal{A},S} w_S^A / w_S^I)$, which we assume it is known. Let OPT be the optimum value and $\mathcal{I}$ denote the input size.

The sample average function is now $\widehat{h}(x) = w^I \cdot x + \sum_{A\in\mathcal{A}} \widehat{p}_A f_A(x)$ where $\widehat{p}_A = \mathcal{N}_A/\mathcal{N}$, with $\mathcal{N}$ being the total number of samples and $\mathcal{N}_A$ being the number of times scenario A is sampled. The sample average problem is $\min_{x\in\mathcal{P}} \widehat{h}(x)$. Observe that the recourse problem for each scenario realized is the same for both the true problem and the sample average one. We will now show that with a polynomially bounded $\mathcal{N}$, $h(.)$ and $\widehat{h}(.)$ satisfy property (A) ("closeness-in-subgradients") with high probability.

To do so, we will use the "nice" subgradient that is calculated from the dual recourse programs. We remind the reader of lemmas 4.7 and 4.8. As the latter lemma was about the stochastic set cover formulation, the subgradient for the above more general program is calculated from a similar formula. More specifically, if (z_A^*) is an optimal solution to the dual of $f_A(x)$ then $d_x = w^I - \sum_A p_A (T^A)^T z_A^*$ is a subgradient of h at x. With similar reasoning, we get that $\widehat{d_x} = w^I - \sum_A \widehat{p}_A (T^A)^T z_A^*$ is a subgradient of $\widehat{h}$ at x, and we also have that $\mathbf{E}[\widehat{d_x}] = d_x$, where the expectation is over the random samples (the reader can also check corollary 4.2). Now, since the vector $w^I - (T^A)^T z_A^*$ lies in $[-\lambda w_S^I, w_S^I]$, we can use the sampling lemma 9.1 to show that property (A) holds. So, we reach the following result:

Theorem 9.1. *For any $\epsilon, \gamma > 0$ ($\gamma \leq 1$), with probability at least $1-\delta$, any optimal solution $\widehat{x}$ to the SAA problem constructed with poly$\Big(\mathcal{I}, \lambda, \frac{1}{\gamma}, \ln\left(\frac{1}{\epsilon}\right), \ln\left(\frac{1}{\delta}\right)\Big)$ samples satisfies $h(\widehat{x}) \leq (1+\gamma)\cdot OPT + 6\epsilon$.*

Proof. We only need to show that property (A) holds with probability at least $1-\delta$ with a polynomial sample size; the rest follows from lemma 9.4. We will use the grid that we have already defined, i.e. we set $N = \log_2\left(\frac{2KR}{\epsilon}\right)$, $\omega = \frac{\gamma}{8N}$, and the extended $\frac{\epsilon V}{8KNR\sqrt{m}}$-grid G of $\mathcal{P}$. Note that $\log\left(\frac{KR}{V}\right)$ is polynomially bounded in the input size. Let $n = |G|$. From lemma 9.2 we get that $n = O\big(N\left(\frac{2R}{\epsilon}\right)^{2m}\big)$. We are now going to use the sampling lemma 9.1. Observe that each random variable lies in $[-\lambda w_S^I, w_S^I]$. Since we need total error probability δ, we can think as follows. We fix a point $x \in G$. We want to calculate an approximate subgradient at this point. Since we work in m dimensions, we need error probability δ'/m for each coefficient of the subgradient, so as the total error probability by union bound is at most δ'. Note that property (A) states that at every point x in the grid we need the "closeness-in-subgradients". Thus, again using union bound, in order to have total error probability δ we must have $\delta' = \delta/n$. Thus, we plug the error probability value $\delta/(mn)$ in the sampling lemma. Also, note that the random variable defined in the lemma as the average of the $\mathcal{N}$ iid random variables corresponds exactly to the formula $w^I - \sum_A \widehat{p}_A (T^A)^T z_A^*$ produced by the SAA problem in order to calculate the subgradient. Thus, using this sampling lemma we achieve exactly what we want.

Plugging things together, we get that $\mathcal{N} = \frac{4(1+\lambda)^2}{3\omega^2}\ln\left(\frac{2mn}{\delta}\right)$ samples are sufficient in order for property (A) to hold with probability at least $(1-\delta)$. We can easily now get that $\mathcal{N} = O\Big(m\lambda^2 \log^2\left(\frac{2KR}{\epsilon}\right)\ln\left(\frac{2KRm}{\epsilon V \delta}\right)/\gamma^2\Big) = \text{poly}\Big(\mathcal{I}, \lambda, \frac{1}{\gamma}, \ln\left(\frac{1}{\epsilon}\right), \ln\left(\frac{1}{\delta}\right)\Big)$.

□

One can convert the above guarantee into a purely multiplicative $(1+z)$-approximation guarantee by setting γ and ϵ appropriately, provided that we have a lower bound on OPT (that is at least inverse exponential in the input size). As shown in chapter 4 and the procedure

ConvOpt presented there, an initial sampling step is sufficient in order to obtain such a lower bound (with high probability). Using this we obtain that (under some mild assumptions) the SAA method returns a $(1+z)$-optimal solution to (2Gen-P) with high probability. This is essentially equivalent to the result stated in theorem 4.3.

9.3.3 Towards multistage problems: the 3-stage stochastic set cover

Before handling k-stage problems, we will focus on the 3-stage Stochastic Set Cover. The ideas presented in this section will be then generalized so as to prove a polynomial bound on k-stage problems.

One way of formulating multistage problems is as exponentially large LP's. However, such a representation does not help, as has already been obvious in the special case of 2-stage Set Cover, studied in section 4.2. Thus, we are going to try a similar approach to the one made in the 2-stage problem, and write our objective function as a function of the first-stage vector.

More formally, in the Stochastic Set Cover Problem, we are given a universe U of n elements and a family $\mathcal{S}$ of m subsets of U, and the set of elements to cover is determined by a probability distribution. In the 3-stage problem this distribution is specified by a 3-level tree. We use A to denote an outcome in stage 2, and (A,B) to denote a stage 3 scenario where A was the stage 2 outcome. Let $\mathcal{A}$ be the set of all stage 2 outcomes, and for each $A \in \mathcal{A}$ let $\mathcal{B}_A = \{B : (A,B) \text{ is a scenario}\}$, that is $\mathcal{B}_A$ is the set of all possible outcomes of stage 3 if the outcome of stage 2 is A. Let p_A and $p_{A,B}$ be the probabilities of outcome A and scenario (A,B) respectively, and let $q_{A,B} = p_{A,B}/p_A$. Note that $\sum_{A\in\mathcal{A}} p_A = 1 = \sum_{B\in\mathcal{B}_A} q_{A,B}$ for every $A \in \mathcal{A}$. We have to cover the (random) set of elements $\mathcal{E}(A,B)$ in scenario (A,B), and we can buy a set S in stage 1, or in stage 2 outcome A, or in scenario (A,B) incurring a cost of w_S^I, w_S^A and $w_S^{A,B}$ respectively. We assume that $\lambda = \max_{S,A\in\mathcal{A},B\in\mathcal{B}_A} \max\left(1, \frac{w_S^A}{w_S}, \frac{w_S^{A,B}}{w_S^A}\right)$ is known.

Let x, y_A and $z_{A,B}$ denote the decisions in stage 1, outcome A and scenario (A,B), respectively. The relaxed 3-stage Set Cover can now be written as follows:

(3SSC-P):

$$\text{min:} \quad h(x) = \sum_S w_S^I x_S + \sum_{A\in\mathcal{A}} p_A f_A(x) \quad \text{subject to } 0 \le x_S \le 1 \quad \text{for all } S,$$

where

$$f_A(x) = \min\Big\{\sum_S w_S^A y_{A,S} + \sum_{B\in\mathcal{B}_A} q_{A,B} f_{A,B}(x, y_A) : y_{A,S} \ge 0 \quad \text{for all } S\Big\}, \qquad \text{(3SSCR-P)}$$

and

$$f_{A,B}(x,y_A) = \min_{z_{A,B}\in\mathbb{R}^m_{\ge 0}} \Big\{\sum_S w_S^{A,B} z_{A,B,S} : \sum_{S:e\in S} z_{A,B,S} \ge 1 - \sum_{S:e\in S}(x_S + y_{A,S}) \quad \forall e \in \mathcal{E}(A,B)\Big\}.$$

In order to get a better understanding of the above program, observe that the recourse problem (3SSCR-P) is a 2-stage Set Cover instance. Let $\mathcal{P} = \{x \in \mathbb{R}^m : 0 \le x_S \le 1, \text{ for all } S\}$ and $OPT = \min_{x\in\mathcal{P}} h(x)$. Note that $\mathcal{P}$ contains a ball of radius $V = 1/2$, and is contained in $B(\mathbf{0}, R)$ where $R = \sqrt{m}$. The sample average problem is parameterized by (i) the sample size $\mathcal{T}_2$ used to estimate probability p_A by the frequency $\widehat{p}_A = \mathcal{T}_{2;A}/\mathcal{T}_2$, and (ii) the number of samples $\mathcal{T}_3$ generated from the conditional distribution of scenarios in $\mathcal{B}_A$ for each A with $\widehat{p}_A > 0$ to estimate $q_{A,B}$ by $\widehat{q}_{A,B} = \mathcal{T}_{3;A,B}/\mathcal{T}_3$. So the total sample size is $\mathcal{T}_2 \cdot \mathcal{T}_3$. The sample average problem is similar to (3SSC-P) with $\widehat{p}_A$ replacing p_A, and $\widehat{q}_{A,B}$ replacing $q_{A,B}$ in the recourse problem $f_A(x)$. We use $\widehat{f}_A(x) = \min_{y_A\ge\mathbf{0}}\Big(w^A \cdot y_A + \sum_{B\in\mathcal{B}_A} \widehat{q}_{A,B} f_A(x, y_A)\Big)$ to denote the sample average recourse problem for outcome A, and $\widehat{h}(x) = w^I \cdot x + \sum_{A \in \mathcal{A}} \widehat{p}_A \widehat{f}_A(x)$ to denote the sample average function.

Observe now that the recourse problems of the true problem and the sampled problem are different for each outcome A. This is a crucial point, as it makes the analysis much more difficult for $k \geq 3$ stages. As already stated, in order to prove the "closeness-in-subgradients" property, we focus on a certain subgradient computed from the solutions of the dual of the recourse problem. Since the recourse problems are different, so are their duals and thus we cannot proceed without modifying these duals. Our goal is to modify them in such a way so that we can prove that solving one dual to optimality is equivalent to solving the other to near-optimality, that is, we need to show an SAA theorem for these two duals. To achieve this, we first formulate the dual as a compact concave maximization problem, then slightly modify the two dual programs and show that the dual objective functions become close in terms of their max-subgradients, and then use lemma 9.5 to obtain the required SAA theorem (for the duals). A max-subgradient of the dual objective function is obtained from the optimal solution of a 2-stage primal problem and we use theorem 9.1 to prove the closeness in max-subgradients of the sample average dual and the true dual. This method can be then applied inductively to prove an SAA bound for a large class of k-stage stochastic LPs.

A parenthesis: the Lagrangian relaxation

We will now introduce the notion of Lagrangian relaxation, from which we formulate the Lagrangian dual. Lagrangian relaxation is a technique well suited for problems where the constraints can be divided into two sets:

- "good" constraints, with which the problem is solvable very easily
- "bad" constraints that make it very hard to solve.

The main idea is to relax the problem by removing the "bad" constraints and putting them into the objective function, assigned with weights (the Lagrangian multipliers). Each weight represents a penalty which is added to a solution that does not satisfy the particular constraint. We will give a simple example in order to get some intuition about this technique. Consider an LP problem $x \in \mathbb{R}^n$, $A \in \mathbb{R}^{m,n}\mathbb{R}^n$, $c \in \mathbb{R}^n$ and $b \in \mathbb{R}^n$ of the following form:

$$\begin{aligned} \text{min:} \quad & c^T \cdot x \\ \text{s.t.:} \quad & Ax \geq b \end{aligned}$$

If we split the constraints in A such that $A_1 \in \mathbb{R}^{m_1,n}$, $A_2 \in \mathbb{R}^{m_2,n}$ and $m1 + m2 = m$ we may write the system:

$$\begin{aligned} \text{min:} \quad & c^T \cdot x \\ \text{s.t.:} \quad & A_1 x \geq b_1 \\ & A_2 x \geq b_2 \end{aligned}$$

We assume now that optimizing over the first set of constraints can be done very easily, whereas adding the "bad" constraints $A_2 x \geq b_2$ makes the problem intractable. Therefore, we introduce a dual variable for every constraint of $A_2 x \geq b_2$. The vector $\lambda \geq \mathbf{0}$ is the vector of dual variables (the Lagrangian multipliers) that has the same dimension as vector b_2. For a fixed $\lambda \geq \mathbf{0}$, consider the relaxed problem

$$Z(\lambda) = \min\{c^T \cdot x + \lambda^T (b_2 - A_2 x) : A_1 x \geq b_1\}.$$

By assumption, we can efficiently compute the optimal value for the relaxed problem with a fixed vector λ. Observe that since $\lambda \geq \mathbf{0}$, we get penalized if we violate the "bad" constraints, and we are also rewarded if we satisfy the constraint strictly. The above system is called the Lagrangian Relaxation of our original problem.

Of particular use is the property that for any λ, the optimal result to the Lagrangian Relaxation problem will be smaller than the optimal result to the original problem, i.e. $Z(\lambda)$ provides lower bounds to the optimal value of the initial problem (weak duality). This is easily provable. Now, if we seek the highest lower bound, we formulate the problem

$$P(\lambda) = \max\{Z(\lambda) : \lambda \geq \mathbf{0}\}.$$

The above program is called the Lagrange dual.

A Lagrangian Relaxation algorithm thus proceeds to explore the range of feasible λ values while seeking to maximize the result returned by the inner Z problem. Each value returned by Z is a candidate lower bound to the problem, the highest of which is kept as the best lower bound. If we additionally employ a heuristic, probably seeded by the values returned by Z, to find feasible solutions to the original problem, then we can iterate until the best lower bound and the cost of the best feasible solution converge to a desired tolerance. The interested reader can refer to [6] for more information on Lagrange duality.

Returning back to our problem, we will formulate the dual of the recourse problem as a Lagrangian dual. Let $f_A(\mathbf{0}; W)$ (respectively $\widehat{f}_A(\mathbf{0}; W)$) denote the recourse problem $f_A(\mathbf{0})$ (respectively $\widehat{f}_A(\mathbf{0})$) with costs $w^A = W$, that is, $f_A(\mathbf{0}; W) = \min_{y_A \geq \mathbf{0}} \big(W \cdot y_A + \sum_{B \in \mathcal{B}_A} q_{A,B} f_{A,B}(\mathbf{0}, y_A)\big)$. To produce the dual, we do the following:

- We add the (redundant) constraints $x_S + y_{A,S} \geq r_S$ to $f_A(x)$.
- We then write the objective function of $f_A(x)$ as $\sum_S w_S^A y_{A,S} + \sum_{B \in \mathcal{B}_A} q_{A,B} f_{A,B}(\mathbf{0}, r)$.
- We finally take the Lagrangian dual of the resulting program by dualizing (i.e. labeling as "bad") only the $x_S + y_{A,S} \geq r_{A,S}$ constraint using $\alpha_{A,S}$ as the Lagrangian multiplier.

The resulting duals of the true and the sample average recourse problems are:

$$LD_A(x) = \max_{\mathbf{0} \leq \alpha_A \leq w^A} l_A(x; \alpha_A) \qquad \text{and} \qquad \widehat{LD}_A(x) = \max_{\mathbf{0} \leq \alpha_A \leq w^A} \widehat{l}_A(x; \alpha_A)$$

where $l_A(x; \alpha_A) = -\alpha_A \cdot x + f_A(\mathbf{0}; \alpha_A)$ and $\widehat{l}_A(x; \alpha_A) = -\alpha_A \cdot x + \widehat{f}_A(\mathbf{0}; \alpha_A)$.

The above formulation will prove really convenient. The first result obtained is strong duality. More formally:

Lemma 9.6. *At any point $x \in \mathcal{P}$ and outcome $A \in \mathcal{A}$, $f_A(x) = LD_A(x)$ and $\widehat{f}_A(x) = \widehat{LD}_A(x)$.*

The proof is omitted. As a note to the reader, for the rest of this chapter, most proofs will be omitted, as there are many intermediate results to be proved in order to reach our goal and moreover, many of the proofs become even more technical than the ones presented in chapter 4, and so we believe that analytical presentation of each proof would discourage the reader. The very interested reader can refer to the paper itself to delve into these technical details. However, we will try to present some of their basic ideas and get some intuition of the results obtained.

We will now use the above lemma to prove that we can computer a nice subgradient from the optimal solution to the dual constructed (we again remind the reader of lemmas 4.7 and 4.8).

Lemma 9.7. *Fix $x \in \mathcal{P}$. Let α_A be a solution to $LD_A(x)$ of value $l_A(x;\alpha_A) \geq (1-\epsilon)LD_A(x) - \epsilon w^I \cdot x - \epsilon'$ for every $A \in \mathcal{A}$. Then, (i) $d = w^I - \sum_A p_A \alpha_A$ is an (ϵ,ϵ')-subgradient of $h(.)$ at x with $\|d\| \leq \lambda \|w^I\|$; (ii) if $\widehat{d}$ is a vector such that $d - \omega w^I \leq \widehat{d} \leq d + \omega w^I$, then $\widehat{d}$ is an $(\epsilon+\omega,\epsilon')$-subgradient of $h(.)$ at x.*

The proof of the above lemma is pretty straightforward. Now, since $\widehat{h}(.)$ is of the same form as $h(.)$, we can use the above lemma to show that $\widehat{d}_x = w^I - \sum_A \widehat{p}_A \widehat{\alpha}_A$ is a subgradient of $\widehat{h}(.)$ at x where $\widehat{\alpha}_A$ is an optimal solution to $\widehat{LD}_A(x)$. The above lemma also shows that both h and $\widehat{h}$ have Lipschitz constant at most $\lambda\|w^I\|$. Observe now that if we prove that any optimal solution to $\widehat{LD}_A(x)$ is a near-optimal solution to $LD_A(x)$, then by using the same approach used in the 2-stage problem, we can prove that the expected value of $\widehat{d}_x$ is an approximate subgradient of $h(.)$, and so, by utilizing the sampling lemma again we will be able to prove the "closeness-in-subgradients" of h and $\widehat{h}$.

However, in our attempt to prove that any optimal solution to $\widehat{LD}_A(x)$ is a near-optimal solution to $LD_A(x)$ we run into some technical difficulties. We could try to argue this by showing that $l_A(x;.)$ and $\widehat{l}_A(x;.)$ are close in terms of their max-subgradients (that is, satisfy property (B)) because $LD_A(x)$ and $\widehat{LD}_A(x)$ are maximization problems, however some problems arise here. A max-subgradient of $l_A(x;.)$ at α_A is obtained from a solution to the 2-stage problem given by $f_A(\mathbf{0};\alpha_A)$ (we will see this in detail in lemma 9.11), and to show closeness in max-subgradients at α_A we need to argue that an optimal solution $\widehat{y}_A$ to $\widehat{f}_A(\mathbf{0};\alpha_A)$ is a near-optimal solution to $f_A(\mathbf{0};\alpha_A)$. The first thing that comes to mind is theorem 9.1 since we now have a 2-stage problem, but this statement need not be true (with a polynomial sample size) since the ratio $\max_S\left(\frac{w_S^{A,B}}{\alpha_{A,S}}\right)$ of the second and first-stage costs in the 2-stage problem $f_A(\mathbf{0};\alpha_A)$, could be unbounded. To tackle this, we modify the feasible region of α_A so as to bound this ratio, and so we consider instead the modified dual problems

$$LD_{A;p}(x) = \max_{pw^I \leq \alpha_A \leq w^A} l_A(x;\alpha_A) \qquad \text{and} \qquad \widehat{LD}_{A;p}(x) = \max_{pw^I \leq \alpha_A \leq w^A} \widehat{l}_A(x;\alpha_A)$$

for a suitable $p \in (0,1)$. More precisely, we should define the feasible region of $LD_{A;p}(x)$ and $\widehat{LD}_{A;p}(x)$ as $\{\alpha_A \in \mathbb{R}^m : p\min\{w^I,w^A\} \leq \alpha_A \leq w^A\}$ to ensure that the feasible region is non-empty; we assume for simplicity for now on that $w^I \leq w^A$. Observe that we have achieved to bound the cost ratio in the 2-stage problem $f_A(\mathbf{0};\alpha_A)$ by $\frac{\lambda^2}{p}$ for any $A \in \mathcal{A}$.

We can now prove the following SAA theorem for the above modified duals.

Lemma 9.8. *For any parameters $\epsilon,\epsilon' > 0$, $p \in (0,1)$, any $x \in \mathcal{P}$, and any outcome $A \in \mathcal{A}$, if we use $\mathcal{T}(\epsilon,p,\epsilon',\delta) = \text{poly}\big(\mathcal{I}, \frac{\lambda}{p\epsilon'}, \ln(\frac{1}{\epsilon}), \ln(\frac{1}{\delta})\big)$ samples to construct the recourse problem $\widehat{f}_A(x)$, then any optimal solution $\widehat{\alpha}_A$ to $\widehat{LD}_{A;p}(x)$ satisfies $l_A(x;\widehat{\alpha}_A) \geq (1-\epsilon')LD_{A;p}(x) - \epsilon' w^I \cdot x - \epsilon$ with probability at least $1-\delta$.*

We now define the modified objective value functions $h_p(x) = w^I \cdot x + \sum_A p_A LD_{A;p}(x)$ and $\widehat{h}_p(x) = w^I \cdot x + \sum_A \widehat{p}_A \widehat{LD}_{A;p}(x)$. As in lemma 9.7, one can show that near-optimal solutions α_A to $LD_{A;p}(x)$ for every $A \in \mathcal{A}$ yield an approximate subgradient of $h_p(.)$ at x. So using lemma 9.8 we can show the closeness in subgradients of $h_p(.)$ and $\widehat{h}_p(.)$, and this will suffice to show that if $\widehat{x}$ minimizes $\widehat{h}(.)$ then it is a near-optimal solution to $h(.)$. Thus we get an SAA bound for our class of 3-stage programs.

More formally, the lemma below proves the closeness in subgradients of $h_p(.)$ and $\widehat{h}_p(.)$ with probability at least $1-\delta$.

Lemma 9.9. *Consider the sample average function generated by taking $\mathcal{N}_2 = \mathcal{T}_2(\omega,\delta) = \frac{16(1+\lambda)^2}{\omega^2}\ln\left(\frac{4m}{\delta}\right)$ samples from stage 2, and $\mathcal{T}\big(\epsilon,p,\frac{\omega}{2},\frac{\delta}{2\mathcal{N}_2}\big)$ samples from stage 3 for each out-*

come A with $\widehat{p}_A > 0$. At any point $x \in \mathcal{P}$, subgradient $\widehat{d}_x$ of $\widehat{h}_p(.)$ is an (ω, ϵ)-subgradient of $h_p(.)$ with probability at least $1 - \delta$.

The proof is omitted. It is quite technical because if we consider the random variable taking the value $w_S - \widehat{\alpha}_{A,S}$ when outcome A is sampled, where $\widehat{\alpha}_A$ is an optimal solution to $\widehat{LD}_{A;p}(x)$, then the random variables corresponding to the different samples from stage 2 are not independent since we always use the same solution $\widehat{\alpha}_A$.

We will also need the result below that relates the values of $h_p(.)$ and $h(.)$ (and, respectively, $\widehat{h}_p(.)$ and $\widehat{h}(.)$).

Lemma 9.10. *For any $x \in \mathcal{P}$, $h_p(x) \leq h(x) \leq h_p(x) + pw^I \cdot x$. Similarly, $\widehat{h}_p(x) \leq \widehat{h}(x) \leq \widehat{h}_p(x) + pw^I \cdot x$.*

Proof. We prove this for $h(.)$ and $h_p(.)$. The second statement is proved identically. The first inequality comes directly from the definition of $h(.)$ and $h_p(.)$. Since we are maximizing over a larger feasible region in $LD_A(x)$, we have that $LD_A(x) \geq LD_{A;p}(x)$ for any x and A, and so $h_p(x) \leq h(x)$. Consider now the optimal solution α^*_A of $l_A(x; .)$, that is $LD_A(x) = l_A(x; \alpha^*_A)$. We now choose an α'_A in the feasible region of $LD_{A;p}$ by shifting α^*_A, so we consider $\alpha'_A = \min(\alpha^*_A + pw^I, w^A)$. We now have $LD_{A;p}(x) \geq l_A(x; \alpha'_A) \geq l_A(x; \alpha^*_A) - pw^I \cdot x$. The last inequality holds because $f_A(\mathbf{0}; \alpha_A)$ is increasing in α_A and $x, w \geq \mathbf{0}$. So, $h_p(x) \geq h(x) - px^I \cdot x$. □

We can state our main theorem now.

Theorem 9.2. *For any $\epsilon, \gamma > 0$ $(\gamma \leq 1)$, one can construct $\widehat{h}$ with* $\text{poly}\Big(\mathcal{I}, \lambda, \frac{1}{\gamma}, \ln\left(\frac{1}{\epsilon}\right), \ln\left(\frac{1}{\delta}\right)\Big)$ *samples, and with probability at least $1 - \delta$ any optimal solution $\widehat{x}$ to $\min_{x \in \mathcal{P}} \widehat{h}(x)$ satisfies $h(\widehat{x}) \leq (1 + 3\gamma) \cdot OPT + 16\epsilon$.*

Outline of the Proof. The proof has two parts. We first show that a near-optimal solution to $\min_{x \in \mathcal{P}} \widehat{h}_p(x)$ yields a near-optimal solution to $\min_{x \in \mathcal{P}} h_p(x)$. To do so, we prove that property (A) holds in the extended-grid that we have already mentioned. Then, we use lemma 9.4 and get that $h_p(\tilde{x}) \leq (1+\gamma)OPT_p + 6\epsilon + 2N\epsilon'$ with high probability, where $\tilde{x}$ is an optimal solution to $\min_{x \in \mathcal{P}} \widehat{h}_p(x)$ and $OPT_p = \min_{x \in \mathcal{P}} h_p(x)$. Since $\widehat{h}_p(\widehat{x}) \leq \widehat{h}_p(\tilde{x}) + pw^I \cdot \tilde{x}$, i.e. $\widehat{x}$ is an approximate minimizer of $\widehat{h}_p$, we use corollary 9.1 and obtain

$$h_p(\widehat{x}) \leq (1 + \gamma)OPT_p + 6\epsilon + 2N(pw^I \cdot \tilde{x} + \epsilon')$$

The second part of the proof is to show that minimizing $h(.)$ and $\widehat{h}(.)$ over $\mathcal{P}$ is roughly the same as approximately minimizing $h_p(.)$ and $\widehat{h}_p(.)$ respectively over $\mathcal{P}$, a fact which explains why the closeness in subgradients of $h_p(.)$ and $\widehat{h}_p(.)$ that we had mentioned before suffices in order to show that if $\widehat{x}$ minimizes $\widehat{h}(.)$ then it is a near-optimal solution to $h(.)$. The bound of lemma 9.10 implies that $(1 - p)h(\widehat{x}) \leq h_p(\widehat{x})$. Also, we have $w^I \cdot \tilde{x} \leq h_p(\tilde{x})$. Now using the bound $OPT_p \leq OPT$ and the bound on $h_p(\tilde{x})$ and plugging ϵ' and p in the above equation, we get that

$$h(\widehat{x}) \leq (1 + 3\gamma)OPT + 16\epsilon.$$

□

In order to get a multiplicative guarantee, we will again need an initial sampling procedure, which works under some mild conditions, that is for every scenario (A, B) with $\mathcal{E}(A, B) \neq \emptyset$, for every $x \in \mathcal{P}$ and $y_A \geq \mathbf{0}$ the total cost $w^I \cdot x + w^A \cdot y_A + f_{A,B}(x, y_A)$ is at least 1. In such cases a sampling procedure similar to the one used for 2-stage programs (we will formally define it in the next section, where we will generalize about k-stage problems) gives a lower bound on

OPT. Thus we obtain a $(1+z)$-optimal solution to (3SSC-P) with the SAA method (with high probability) using polynomially many samples.

One last thing now remains, how are subgradients are computed from the duals we have formulated. As observed, we only need to compute subgradients of the functions $h_p(x)$ and $\widehat{h}_p(x)$. The following lemma gives the solution.

Lemma 9.11. *Fix $x \in \mathcal{P}$ and $\alpha_A \in \mathcal{D}_A$. Let $\omega' = \frac{\omega}{\lambda}$. If y_A is a solution to $f_A(\mathbf{0};\alpha_A)$ of value $g(\alpha_A;y_A) \leq (1+\omega')f_A(\mathbf{0};\alpha_A)+\epsilon'$, then $d = y_A - x$ is an $(\omega, \omega w^I \cdot x + \epsilon')$-max-subgradient of $l_A(x;.)$ at α_A.*

9.3.4 A class of solvable 3-stage programs

The analysis presented above naturally extends to a wide class of 3-stage programs. We use the same notation as above and consider the following class of 3-stage problems.

(3Gen-P):

$$\text{min:} \quad h(x) = w^I \cdot x + \sum_{A\in\mathcal{A}} p_A f_A(x) \quad \text{subject to} \quad x \in \mathcal{P} \subseteq \mathbb{R}^m_{\geq 0},$$

where

$$f_A(x) = \min_{y_A \in \mathbb{R}^m_{\geq 0}} \Big\{ w^A \cdot y_A + \sum_{B\in\mathcal{B}_A} q_{A,B} f_{A,B}(x,y_A) : T^A y_A \geq j^A - T^A x \Big\}, \qquad \text{(3Rec-P)}$$

and

$$f_{A,B}(x,y_A) = \min_{\substack{z_{A,B}\in\mathbb{R}^m_{\geq 0}\\ s_{A,B}\in\mathbb{R}^n_{\geq 0}}} \Big\{ w^{A,B} \cdot z_{A,B} + c^{A,B} \cdot s_{A,B} : \\ D^{A,B} s_{A,B} + T^{A,B} z_{A,B} \geq j^{A,B} - T^{A,B}(x+y_A) \Big\}.$$

where for every outcome $A \in \mathcal{A}$ and scenario (A,B), (a) $T^A, T^{A,B} \geq 0$; (b) for every $x \in \mathcal{P}$, and $y_A \geq \mathbf{0}$, $0 \leq f_A(x), f_{A,B}(x,y_A) < +\infty$. Let $\lambda = \max_{S,A\in\mathcal{A},B\in\mathcal{B}_A} \max\Big(1, \frac{w_S^A}{w_S}, \frac{w_S^{A,B}}{w_S^A}\Big)$. As before, we assume that λ is known, that $\mathcal{P} \subseteq B(\mathbf{0},R)$ and that $\mathcal{P}$ contains a ball of radius $V \leq 1$, where $\ln(R/V)$ is polynomially bounded. Further we assume that for any $x \in \mathcal{P}$ and any $A \in \mathcal{A}$, the feasible region of $f_A(x)$ can be restricted to $B(\mathbf{0},R)$ without affecting the solution quality, that is, there is an optimal solution to $f_A(x)$ lying in $B(\mathbf{0},R)$. These assumptions are fairly mild and unrestrictive; in particular, they hold trivially for the fractional relaxations of many combinatorial optimization problems. Let OPT be the optimum value and $\mathcal{I}$ be the input size.

The sample average problem is of the same form as (3Gen-P), where p_A and $q_{A,B}$ are replaced by their estimates $\widehat{p}_A$ and $\widehat{q}_{A,B}$ respectively, the frequencies of occurrence of outcome A and scenario (A,B) in the appropriate sampled sets. Let $\widehat{h}(x) = w^I \cdot x + \sum_{A\in\mathcal{A}} \widehat{p}_A \widehat{f}_A(x)$ denote the sample average function where

$$\widehat{f}_A(x) = \min_{y_A \geq \mathbf{0}} \Big\{ w^A \cdot y_A + \sum_{B\in\mathcal{B}_A} \widehat{q}_{A,B} f_{A,B}(x,yA) : T^A y_A \geq j^A - T^A x \Big\} \qquad \text{(3SARec-P)}$$

is the sample average recourse problem.

Let $f_A(\mathbf{0};W)$ (respectively $f_A(\mathbf{0};W)$) denote the recourse problem (3Rec-P) (respectively (3SARec-P)) with $x = \mathbf{0}$ and costs $w^A = W$. The dual of the recourse problem is formulated as before, $LD_A(x) = \max_{\mathbf{0}\leq\alpha_A\leq w^A} l_A(x;\alpha_A)$ where $l_A(x;\alpha_A) = -\alpha_A \cdot x + f_A(\mathbf{0};\alpha_A)$. We use $\widehat{LD}_A(x)$ and $\widehat{l}_A(x;\alpha_A)$ to denote the corresponding quantities for the sample average problem.

The analysis is almost identical, except the lemma that proves strong duality in the new dual representation. At the end, we manage to reach the following result.

Theorem 9.3. *For any parameters $\epsilon, \gamma > 0$ ($\gamma \leq 1$), one can construct the sample average problem $\widehat{h}$ using $\operatorname{poly}\Big(\mathcal{I}, \lambda, \frac{1}{\gamma}, \ln\left(\frac{1}{\epsilon}\right), \left(\frac{1}{\delta}\right)\Big)$ samples so that, with probability at least $1-\delta$, any optimal solution $\widehat{x}$ to $\widehat{h}$ has value $h(\widehat{x}) \leq (1+3\gamma) \cdot OPT + 16\epsilon$.*

In the general case we do not have multiplicative guarantee. However, for the subclass of (3Gen-P) where the recourse problem $f_A(x)$ does not have any constraints, an initial sampling allows us to get a $(1+z)$-guarantee.

9.3.5 The SAA bound for k-stage programs

We will now define a class of k-stage programs and see that a polynomial sample size suffices to give an approximation scheme. We consider k to be a fixed constant and not part of the input. The running time of our algorithm will be *exponential* to k. The description is the same as in §7 of [43].

In order to better understand our approach, we should bring back in mind the visualization of a k-stage problem as a tree. We start at the root r of this tree at level 1, which represents the first-stage. Let $level(i)$ denote the set of nodes at level i, so $level(1) = \{r\}$. Each such node u represents an outcome in stage i and its ancestors correspond to the outcomes in the previous stages; so node u represents a particular evolution of the uncertainty through stages $1, ..., i$. At a leaf node, the uncertainty has completely resolved itself and we know the input precisely. A scenario will always refer to a stage k outcome, that is, a leaf of the tree. The goal is to choose the first stage elements so as to minimize the total expected cost, i.e., $\sum_{i=1}^{k} \mathbf{E}[\text{stage } i \text{ cost}]$ where the expectation is taken over all scenarios.

As already mentioned, each node at level i corresponds to a remaining $(k-i+1)$-stage problem. This observation will help us define our problem recursively. Let $path(u)$ be the set of all nodes (including u) on u's path to the root. Let $child(u)$ be the set of all children of u; this is the set of possible outcomes in the next stage given that u is the current outcome. Let p_u be the probability that outcome u occurs, and q_u be the conditional probability that u occurs given the outcome in the previous stage. We do not assume anything about the distribution, and it can incorporate various correlation effects from previous stages. Note that $p_u = \prod_{v \in path(u)} q_v$. Clearly we have $p_r = q_r = 1$, for any i we have $\sum_{v \in level(i)} p_v = 1$, and for any node u, $\sum_{v \in child(u)} q_v = 1$.

We use y_u to refer to the decisions taken in outcome u and w^u to denote the costs in outcome u; thus the costs may depend on the history of outcomes in the previous stages. Note that y_u may only depend on the decisions in the previous outcomes, that is, on the y_v's where $v \in path(u)$. For convenience we use $x \equiv y_r$ to denote the first-stage decisions, and w^I to denote the first-stage costs. We now consider the following generic k-stage linear program.

$$f_{k,r} = \min \quad h(x) = w^I \cdot x + \sum_{u \in child(r)} q_u f_{k-1,u}(x) \quad \text{subject to} \quad x \in \mathcal{P} \subseteq \mathbb{R}^m_{\geq 0} \qquad \text{(kGen-P)}$$

where $f_{k-1,u}(x)$ gives the expected cost of stages $2, ..., k$ given the first-stage decision x and when u is the stage 2 outcome. Thus $f_{k-1,u}(x)$ is the cost of the $(k-1)$-stage problem that is obtained when u is the second-stage outcome, and x is the first-stage decision. In general, consider an outcome $u \in level(i)$ and let $v \in level(i-1)$ be its parent. Let $\mathbf{y}_v = (y_r, ..., y_v)$, where $\{r, ..., v\} = path(v)$, denote the collective tuple of decisions taken in the previous stages; for the root r, $\mathbf{y}_r \equiv y_r \equiv x$. The function $f_{k-i+1,u}(\mathbf{y}_v)$ is (the cost of) the $(k-i+1)$-stage stochastic program that determines the expected cost of stages $i, ..., k$ given the decisions in the previous stages $\mathbf{y}_v$, and when u is the outcome in stage i. We believe that the above description is quite clear and matches the tree representation of the problem. We now define function $f_{k-i+1,u}(\mathbf{y}_v)$ recursively as

$$f_{k-i+1,u}(\mathbf{y}_v) = \min\big\{w^u \cdot y_u + \sum_{u' \in \mathit{child}(u)} q_{u'} f_{k-1,u'}(\mathbf{y}_v, y_u) : y_u \in \mathbb{R}^m_{\geq 0},\ T^u y_u \geq j^u - \sum_{t \in \mathit{path}(v)} T^u y_t\big\},$$

for a non-leaf node $u \in \mathit{level}(i)$, $2 \leq i \leq k$. For a leaf u at level k,

$$f_{1,u}(\mathbf{y}_v) = \min\big\{w^u \cdot y_u + c^u \cdot s_u : y_u \in \mathbb{R}^m_{\geq 0}, s_u \in \mathbb{R}^n_{\geq 0},\ D^u s_u + T^u y_u \geq j^u - \sum_{t \in \mathit{path}(v)} T^u y_t\big\}.$$

The variables s_u appearing in $f_{1,u}(.)$, capture the fact that at a scenario u when we know the input precisely, one might need to make some additional decisions. We require that (a) $T^u \geq \mathbf{0}$ for every node u; (b) $0 \leq f_{k-i+1,u}(\mathbf{y}_v) < \infty$ for every node $u \in \mathit{level}(i)$ with parent v, and feasible decisions $\mathbf{y}_v$ - this ensures that the primal problem $f_{k-i+1,u}(\mathbf{y}_v)$ and its dual are feasible for every feasible $\mathbf{y}_v$; and (c) there are R and $V \leq 1$ with $ln(R/V)$ polynomially bounded such that for every internal node u, the feasible region $f_{k-i+1,u}(\mathbf{y}_v)$ contains a ball of radius V, and can be restricted to $B(\mathbf{0}, R)$ without affecting the solution quality (so $\mathcal{P} \subseteq B(\mathbf{0}, R)$ and contains a ball of radius V); that is, for each $f_{k-i+1,u}(\mathbf{y}_v)$ there is some optimal solution y^*_u such that $\|y^*_u\| \leq R$. Let $\mathcal{I}$ denote the input size, λ be the ratio $\max\big(1, \max_{v,u \in \mathit{child}(v),S} \frac{w^u_S}{w^v_s}\big)$, and K be the Lipschitz constant of $h(.)$. Define $OPT = f_{k,r}$.

The sample average problem is of the same form as (kGen-P), where the probability q_u is replaced by its estimate $\widehat{q}_u$, which is the frequency of occurrence of outcome u in the appropriate sampled set. It is constructed as follows: we sample $\mathcal{T}_2$ times from the entire distribution and estimate the probability q_u of a node $u \in \mathit{level}(2)$ by its frequency of occurrence $\widehat{q}_u = \mathcal{T}_{2;u}/\mathcal{T}_2$; for each u such that $\widehat{q}_u > 0$, we sample $\mathcal{T}_3$ times from the conditional distribution of scenarios in the tree rooted at u and estimate the probability $q_{u'}$ for each $u' \in \mathit{child}(u)$ by the frequency $\widehat{q}_{u'} = \mathcal{T}_{3;u'}/\mathcal{T}_3$. We continue this way, sampling for each node u such that $\widehat{q}_u > 0$, the leaves of the tree rooted at u to estimate the probabilities of the children of u, till we reach the leaves of the distribution tree. Let $\widehat{p}_u = \prod_{v \in \mathit{path}(u)} \widehat{q}_v$ denote the probability of occurrence of outcome u in the sample average problem. We use $\widehat{f}_{k,r}$ to denote the k-stage sample average problem; correspondingly for node $u \in \mathit{level}(i)$ (where $\widehat{p}_u > 0$) with parent v, $\widehat{f}_{k-i+1,u}(\mathbf{y}_v)$ is the $(k-i+1)$-stage program in the sample average problem that determines the expected cost of stages $i, ..., k$ when outcome u occurs and given the decisions $\mathbf{y}_v$ in the previous stages. Note that for a leaf u, $f_{1,u}(\mathbf{y}_v)$ is simply a (1-stage) deterministic linear program, so $\widehat{f}_{1,u}(\mathbf{y}_v) = f_{1,u}(\mathbf{y}_v)$. Let $\widehat{h}(x)$ be the objective function of the k-stage sample average program, so $\widehat{f}_{k,r} = \min_{x \in \mathcal{P}} h(x)$.

We are now in position to extend the argument used in the previous sections inductively to prove an SAA bound for the k-stage problem $f_{k,r}$. We can show that assuming inductively a polynomial SAA bound $\mathcal{N}_{k-1}$ for the $(k-1)$-stage problem $f_{k-1,r}$, one can construct the sample average problem $\widehat{f}_{k,r}$ with a sufficiently large polynomial sample size, so that, with high probability, any optimal solution to $\widehat{f}_{k,r}$ is a near-optimal solution to $f_{k,r}$. Combined with the previous results which provide the base case in this argument, this establishes a polynomial SAA bound for k-stage programs of the form (kGen-P).

We won't go into any details about the proof. We will only state that again our goal is to prove the "closeness-in-subgradients" property. Finally, we reach the result:

Theorem 9.4. *For any $\epsilon, \gamma > 0$, $(\gamma < 1)$, with probability at least $1-\delta$, any optimal solution $\widehat{x}$ to the k-stage sample average problem constructed using* $\text{poly}\Big(\mathcal{I}, \frac{\lambda}{\gamma}, \ln\big(\frac{1}{\epsilon\delta}\big)\Big)$ *samples satisfies $h(\widehat{x}) \leq (1+\gamma) \cdot f_{k,r} + \epsilon$.*

To obtain a multiplicative guarantee, we must again consider a subclass of the above class of problems, where the recourse problem $f_{k-i+1,u}(x, \mathbf{y}_u)$ does not have any constraints. The desired subclass is the following:

$$g_{k,r} = \min \quad h(x) = w^I \cdot x + \sum_{u \in \mathit{child}(r)} q_u g_{k-1,u}(x) \quad \text{subject to} \quad x \in \mathcal{P} \subseteq \mathbb{R}^m_{\geq 0} \qquad \text{(kSub-P)}$$

where

$$g_{k-i+1,u}(\mathbf{y}_v) = \min \big\{ w^u \cdot y_u + \sum_{u' \in \mathit{child}(u)} q_{u'} g_{k-1,u'}(\mathbf{y}_v, y_u) : y_u \in \mathbb{R}^m_{\geq 0}, \big\},$$

$$\text{for } u \in \mathit{level}(i), 2 \leq i \leq k,$$

and

$$g_{1,u}(\mathbf{y}_v) = \min \big\{ w^u \cdot y_u + c^u \cdot s_u : y_u \in \mathbb{R}^m_{\geq 0}, s_u \in \mathbb{R}^n_{\geq 0},\ D^u s_u + T^u y_u \geq j^u - \sum_{t \in \mathit{path}(v)} T^u y_t \big\}.$$

We once again make the usual assumptions, that (a) $x = \mathbf{0}$ lies in $\mathcal{P}$, and (b) for every scenario u with parent v, either $g_{1,u}(\mathbf{y}_v)$ is minimized by setting $y_t = \mathbf{0}$ for all $t \in \mathit{path}(v)$, or the total cost $\sum_{t \in \mathit{path}(u)} w^t y^t + c^u s^u \geq 1$ for any feasible decisions $(\mathbf{y}_u, s_u)$. For example, for the 3-stage set cover problem considered before, (a) just requires that we are allowed to not pick any set in the first-stage, (b) is satisfied if the total cost incurred in every scenario (A, B) with $\mathcal{E}(A, B) \neq \emptyset$ is at least 1. Under these assumptions, we show that we can sample initially to detect if OPT is large.

We can now reach the following lemma.

Lemma 9.12. *By sampling $M = \lambda^k \ln\left(\frac{1}{\delta}\right)$ times, one can detect with probability at least $1 - \delta$ ($\delta < 1/2$), that either $x = \mathbf{0}$ is an optimal solution to (kSub-P), or that $OPT \geq \frac{\delta}{M}$.*

Thus, we can convert the above guarantee to a purely multiplicative $(1 + z)$-guarantee.

9.4 An overview of our results

We will now explain in brief what we have actually achieved with the analysis in the previous section as regards combinatorial problems. The polynomial size sample bounds that we have proved allow us to solve the SAA problem instead of the true problem, and hope that the solution reached will be a good solution for the true problem. The SAA problem can be seen as a polynomial size LP, and thus it is easily solvable. So, obtaining a good solution vector x for the first-stage, we can proceed with rounding techniques and convert it to an integer one. As the problem progresses, in each stage i we have a $(k - i + 1)$-stage problem, and thus we can use the SAA method repeatedly to solve each such problem and then convert the fractional solution into an integer one. An important note here is that the above description's running time depends on k, and it is polynomial only in the case where k is considered constant and is not part of the input. Otherwise, we have an exponential running time on k (observe that the sample size depends exponentially on k).

We now proceed to see what guarantees we can get with deterministic and randomized rounding techniques for covering and facility location problems.

9.5 Applications

9.5.1 Covering Problems

We believe that the generalization of Set Cover and Vertex Cover to their k-stage variants is pretty clear. The target set of elements to cover is determined by a probability distribution, and becomes known after a sequence of k stages. Using an extension of the rounding theorem 4.1, we can show that one can use a p-approximation algorithm for the deterministic analogue, where the guarantee is with respect to its natural LP relaxation, to round any fractional solution to the k-stage problem to an integer solution losing a factor of p because of each stage. Since there are k stages, we have a total loss factor kp, and using this with the SAA algorithm described above, one gets an $(kp+\epsilon)$-approximation algorithm (note that we have a multiplicative guarantee since we deal with covering problems, which belong in the suitable subclass discussed above). As stated above, in order to compute the decisions in a stage i outcome, we solve a $(k-i+1)$-stage problem and round the solution. All these give the following results.

Theorem 9.5. *Using the SAA theorem 9.4 in conjunction with the rounding theorem 4.1 and the greedy algorithm [8], we get (with high probability) a $(k\ln n+\epsilon)$-approximation algorithm for the k-stage Stochastic Set Cover Problem.*

Theorem 9.6. *In a similar way as described in theorem 9.5, we get (with high probability) a $(2k+\epsilon)$-approximation algorithm for the k-stage Stochastic Vertex Cover.*

We will now see that the randomized rounding suggested by Srinivasan in [39] gives better results than these presented above, and actually manages to weigh the multiplicative dependence of the approximation on the number of stages. We will at first study the Set Cover problem, and then generalize to a class of covering problems.

Let $U=\{e_1,\dots,e_n\}$ be our universe and $\mathcal{S}=\{S_1,\dots,S_m\}$ the family of subsets of U. Let $x^*_l=(x^*_{1,l},x^*_{2,l},\dots,x^*_{m,l})$ be the optimal fractional solution of stage l. We suggest the following randomized rounding approach: for a suitable $t\geq 1$ and independently for all (j,l) set $x'_{j,l}=tx^*_{j,l}$ and define the rounded value $y_{j,l}$ to be

- $\lceil x'_{j,l}\rceil$, with probability $x'_{j,l}-\lfloor x'_{j,l}\rfloor$,
- $\lfloor x'_{j,l}\rfloor$, with probability $1-(x'_{j,l}-\lfloor x'_{j,l}\rfloor)$.

Observe that the above randomized rounding "tends" to round in the integer value that is closer to the real value, no matter whether it is the floor or the ceil of the real value. Also, note that $\mathbf{E}[y_{j,l}]=x'_{j,l}$. We now set $t=\ln n+\psi(n)$, where $\psi(n)$ is an arbitrarily slowly growing function of n such that $\lim_{n\to\infty}\psi(n)=\infty$, and run the above rounding scheme. Consider any finally revealed element e_i, and let E_i be the event that our rounding leaves this element uncovered. Let A_i be the family of sets that contain e_i. We should note that the fractional solution satisfies $\sum_{j\in A_i,l}x^*_{j,l}\geq 1$. Now, if $x'_{j,l}\geq 1$ for some pair $(j\in A_i,l)$, then $y_{j,l}\geq 1$ with probability 1, and, so e_i is covered in such cases. Otherwise, we have that $x'_{j,l}<1$ for all pairs $(j\in A_i,l)$. In this case, we have

$$\begin{aligned}
\Pr[E_i] &= \prod_{j \in A_i, l} \Pr[y_{i,l} = 0] \\
&= \prod_{j \in A_i, l} (1 - x'_{j,l}) && (\text{since } \lfloor x'_{i,j} \rfloor = 0) \\
&= \prod_{j \in A_i, l} (1 - t x^*_{j,l}) \\
&\leq \exp\Big(- \sum_{j \in A_i, l} t \cdot x^*_{j,l} \Big) && (\text{since } e^{-x} \geq 1 - x) \\
&\leq \exp(-t) && \Big(\text{since } \sum_{j \in A_i, l} x^*_{j,l} \geq 1\Big) \\
&= e^{-\psi(n)}/n \\
&= o(1/n).
\end{aligned}$$

Thus, applying a union bound over the (at most n) finally revealed elements e_i, we see that the probability that at least one is not covered is $o(1)$. We now turn to the expected cost incurred in each stage. In each stage l, we have $\mathbf{E}[c^T \cdot y] = c^T \cdot x' = t(c^T \cdot x^*)$. So the expected total cost EXP of the above process, conditioned on the event that all elements are covered is

$$EXP \leq \frac{t \cdot OPT}{1 - o(1)} = (1 + o(1)) \cdot t \cdot OPT,$$

where OPT is the cost of the optimal solution. Thus, we get an $(1 + o(1)) \cdot \ln n$-approximation algorithm. However, there is a crucial detail that the reader must have noticed in our analysis above. The above algorithm does not guarantee success, i.e. we are not sure if at the end we will have a feasible solution to our problem. Such algorithms are known as Monte Carlo algorithms. This is the first Monte Carlo algorithm we present, as all the previous algorithms presented were randomized in the sense that their performance guarantee was probabilistic, while the feasibility of the solution was guaranteed with probability 1. Note also that is is not clear how to eliminate such a possibility of failure without losing much in the approximation guarantee.

The above analysis can be extended to general covering integer programs (CIP's). Such a (CIP) can be seen as a "hidden" covering problem of the form "minimize $c^T \cdot x$ subject to $Ax \geq b$, with all variables in x be non-negative integers". This program, as well as a feasible fractional solution x^* is revealed to us in k stages as follows. In each stage l $(1 \leq l \leq k)$, we are given the l^{th}-stage fractional values $\{x^*_{j,l} : 1 \leq j \leq m\}$ of the variables, along with their columns in the coefficient matrix A, and their coefficient in the objective function c.

So, we finally get the following theorem.

Theorem 9.7. *We obtain randomized (Monte Carlo) t-approximation algorithms for k-stage stochastic CIP's for fixed k, with running time polynomial for any fixed k and t independent of k. More specifically, (i) for general CIP's with the linear system scaled so that all entries of the matrix A lie in $[0,1]$ and $\min_i b_i = B$, we have $t = 1 + O\Big(\min\Big\{\frac{\ln n}{B}, \sqrt{\frac{\ln n}{B}}\Big\}\Big)$. (ii) For Set Cover with element-degree (maximum number of given sets containing any element of the ground set) at most b, we have $t = b + \epsilon$, where ϵ can be $\mathcal{I}^{-C}$ with $\mathcal{I}$ being the input size and $C > 0$ being any constant.*

Note 9.1. *The "$+\epsilon$" term appears in part (ii) since the fractional solution obtained by the SAA method is an $(1 + \epsilon)$-approximation to the actual LP. We do not mention this term in part (i), by absorbing it into the big-Oh notation.*

9.5.2 The Metric Uncapacitated Facility Location Problem

Adapting the rounding scheme of Swamy and Shmoys ([38]), we can get the following result.

Theorem 9.8. *There is a randomized* $1858(k-1)+1.52+\epsilon = O(k)$*-approximation algorithm for the* k*-stage stochastic MUFLP.*

Chapter 10

Conclusion

10.1 Summary - Open questions

Reaching the end of this text, we hope that the reader has gained some understanding of the challenging field of stochastic combinatorial optimization and the various techniques utilized in order to come up with efficient algorithms. We have presented general methods that can be applied in many different problems. However, regarding applications, our attention was turned mostly to the Rooted Steiner Tree, Set Cover, Vertex Cover and Facility Location Problems. Much work has been done concerning other problems, especially 2-stage ones. These include the stochastic versions of the bin-packing [21, 35], the unrooted Steiner Tree [16], Steiner Forest [15] and minimum spanning tree [11], among others.

A number of interesting questions still remains open, especially as regards multistage problems. As already seen, all the approximation algorithms obtained for k-stage problems depend on k, which is considered to be fixed and not part of the input. For both the Boosted Sampling Framework of Gupta etc. and the SAA method of Swamy and Shmoys, the approximation ratios obtained depend on k. Removing this dependence on k is an open problem. For covering and facility location problems, it should be possible to obtain a guarantee for the k-stage problem that almost matches the deterministic guarantee, since one can show that the integrality gap of the k-stage LP, for Set Cover, Vertex Cover, and Facility Location, is close to that of the deterministic LP. We remind the reader here that we have actually achieved to remove such dependence for a large class of covering problems (see section 9.5.1).

Furthermore, both [18] and [43] require a sample size that is exponential in k. A very interesting and challenging problem is to obtain bounds, for linear or integer programs, that are polynomial in k. This may not be possible in the black-box model, but it would be interesting to prove such a bound even for distributions where the different stages are independent. Such results have been obtained in the setting of stochastic inventory control problems in [27] and (with a stronger black box) in [26].

Moreover, it would be interesting to see if the SAA method of Charikar etc. [7] can be extended for multistage problems. As their 2-stage approach actually proves a reduction from the black-box to the polynomial-scenario model for a wide class of integer programs, it would be very useful to obtain such a result for the multistage framework.

There are also interesting variants of stochastic problems that offer many challenges. The simple stochastic recourse model measures the expected cost associated with the first-stage decisions, but often in applications one is also interested in the risk associated with the first-stage decisions, where risk is some measure of the variability (e.g., variance) in the (random) cost incurred in later stages. Gupta etc. in [19] for example consider the use of budgets that bound the cost of each scenario, as a means of guarding against (one-sided) risk, but their results are limited to 2-stage problems with polynomial-scenario distributions. It would be interesting to explore stochastic models that incorporate risk, while allowing for a broader class of distributions (with exponentially many scenarios). The most extensive work in this field of risk-aversion, and the first that we are aware of that gives results in the black-box model is the work of Swamy [41].

Another research avenue, which brings us closer to Markov Decision Problems, is to investigate problems where the uncertainty is affected by the decisions taken. Throughout this thesis we have dealt only with problems in which the underlying distributions are conditionally independent of the recourse actions taken in the previous stages. However, there are problems where the assumption of such independence is not always accurate. Stochastic scheduling problems, where the scheduling decisions interact with the evolution of the random job sizes, provide a fertile ground for such problems.

10.2 Relation to Leasing Problems

The last thing we will talk about is an interesting connection of stochastic problems with another class of problems, known as leasing problems, proposed by Anthony and Gupta [1]. To understand what leasing problems are about, we start with a description taken from [1]. The reader can also look at the result of Meyerson concerning the Parking Permit Problem [31], or for a more thorough discussion of the subject in [25].

Consider a network design problem. Traditional network design problems require us to make decisions about how to send data, and how to provision bandwidth on various links of the network. A standard feature in most models for network design that have been considered, and in the algorithms that have been developed, has been the permanence of the bandwidth allocation—and this has been true even in cases where demands arrive online: once some amount of bandwidth is allocated on an edge, this bandwidth can be used at any time in the future (perhaps by paying some additional incremental "routing cost" per unit of flow). Some works have also considered the question of buying versus renting, but the simplifying assumption again has been that buying gives permanent access to the commodity. But what if we are allowed only to *lease* bandwidth on the links of the network for fixed lengths of time: which leases on which network links should we obtain over time to satisfy our demands, which vary over time?

Given a situation with multiple lease lengths, it is natural to assume that a longer lease is a cheaper one (per day), and that we pay more dearly for the flexibility of short-term leases. Hence, if our traffic consists of some stable parts and other bursty parts, we can use long-term leases to satisfy the stable traffic, and the short term leases to handle the more volatile demands: a clever leasing strategy can reduce costs substantially over a naive one. Note that solving this problem requires us to simultaneously perform clustering over space (in order to figure out which edges to allocate bandwidth on) and over time (to figure out which traffic is stable and requires longer leases, and which is bursty and is best served by shorter leases).

The question of finding good leasing strategies is relevant in the context of other problems as well: in planning for demands arriving over multiple periods in classical facility location problems, one might want to lease warehouses/plants for varying lengths of time. Moreover, the idea that leases of varying lengths are available is fairly natural: even in situations where there is a standard lease length, the presence of a secondary market for reselling or sub-letting might naturally give rise to situations with multiple lease lengths.

The result that Anthony and Gupta reached shows that leasing problems with k lease types can be reduced to k-stage stochastic optimization problems. This immediately allows one to use any algorithm obtained in the multistage optimization framework to solve the corresponding leasing variants of the problems. However, although it seems that stochastic problems are more difficult than leasing problems, it is still an open question whether the reverse reduction is possible or not.

Appendices

Appendix A

Some algebra

We will present here some algebraic notions that we have used in our text. The presentation should in no case be considered complete.

Definition A.1. The epigraph of a function $f : \mathbb{R}^n \to \mathbb{R}$ is the set of points lying on or above its graph:

$$\mathbf{epi}\, f = \{(\mathbf{x}, \mu) \,:\, \mathbf{x} \in \mathbb{R}^n,\, \mu \in \mathbb{R},\, \mu \geq f(\mathbf{x})\} \subseteq \mathbb{R}^{n+1}$$

Definition A.2. A set $X \subseteq \mathbb{R}^n$ is convex if for every two points $\mathbf{x}, \mathbf{y} \in X$ it also contains the straight line segment connecting $\mathbf{x}$ and $\mathbf{y}$. Expressed differently, for every $\mathbf{x}, \mathbf{y} \in X$ and every $t \in [0, 1]$ we have $t\mathbf{x} + (1 - t)\mathbf{y} \in X$.

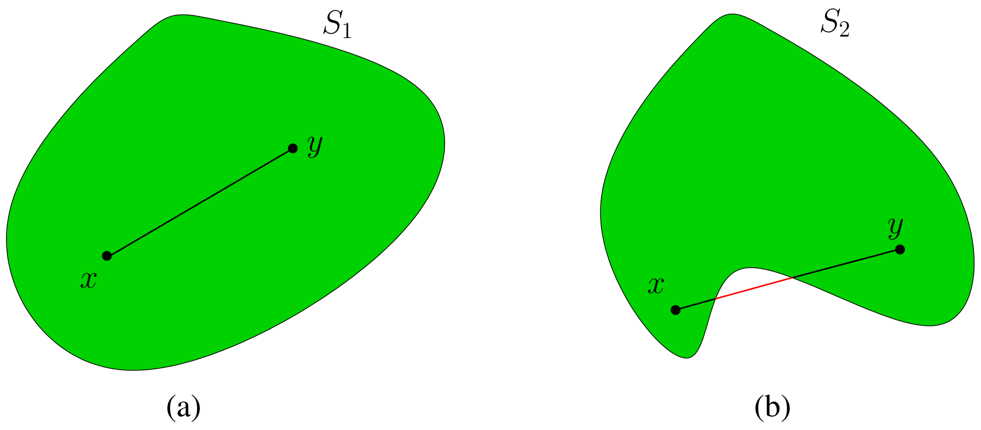

Figure A.1: (a) S_1 is convex, (b) S_2 is non-convex

If S is a convex set, then, for any $\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_r \in S$, and any non-negative numbers $\lambda_1, \lambda_2, ..., \lambda_r$ such that $\lambda_1 + \lambda_2 + ... + \lambda_r = 1$, the vector $\sum_{k=1}^{r} \lambda_k \mathbf{u}_k$ is in S. A vector of this type is known as a convex combination of $\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_r$.

The collection of convex subsets of a vector space has the following properties:

1. The empty set and the whole vector-space are convex.
2. The intersection of any collection of convex sets is convex.
3. The union of a non-decreasing sequence of convex subsets is a convex set.

(For the preceding property of unions of non-decreasing sequences of convex sets, the restriction to nested sets was important: the union of two convex sets need not be convex.)

Definition A.3. Let X be a convex subset of some vector space. A function $f : X \to \mathbb{R}$ is called convex if for every $\mathbf{x}, \mathbf{y} \in X$ and every $t \in [0, 1]$ we have $f(t\mathbf{x} + (1 - t)\mathbf{y}) \leq tf(\mathbf{x}) + (1 - t)f(\mathbf{y})$.

An alternative definition for a convex function is this:

Definition A.4. A function is convex if its epigraph is a convex set.

Definition A.5. Let $X \subseteq \mathbb{R}^n$ be a set. The convex hull of X is the intersection of all convex sets that contain X. Thus it is the smallest convex set containing X, in the sense that any convex set containing X also contains its convex hull. Algebraically, it equals the set

$$\tilde{C} = \big\{ \sum_{i=1}^{m} t_i \mathbf{x}_i : m \geq 1, \mathbf{x}_1, ..., \mathbf{x}_m \in X, t_1, ..., t_m \geq 0, \sum_{i=1}^{m} t_i = 1 \big\}$$

of all convex combinations of finitely many points of X.

Definition A.6. Let V be a vector space over a field K, and let A be a nonempty set. Now define addition $p + \mathbf{a} \in A$ for any vector $\mathbf{a} \in V$ and element $p \in A$ subject to the conditions:

1. $p + \mathbf{0} = p$.
2. $(p + \mathbf{a}) + \mathbf{b} = p + (\mathbf{a} + \mathbf{b})$.
3. For any $q \in A$, there exists a unique vector $\mathbf{a} \in V$ such that $q = p + \mathbf{a}$.

Here, $\mathbf{a}, \mathbf{b} \in V$. Note that (1) is implied by (2) and (3). Then A is an affine space and K is called the coefficient field.

Intuitively, an affine space is what is left of a vector space after you have "forgotten" which point is the origin.

Definition A.7. An affine transformation (or affine map) between two vector spaces (strictly speaking, two affine spaces) consists of a linear transformation followed by a translation:

$$x \mapsto Ax + b.$$

Definition A.8. A hyperplane in $\mathbb{R}^n$ is an affine subspace of dimension $n - 1$. In other words, it is the set of all solutions of a single linear equation of the form

$$a_1x_1 + a_2x_2 + ... + a_nx_n = b$$

where $a_1, a_2, ..., a_n$ are not all 0.

A hyperplane divides $\mathbb{R}^n$ into two **half-spaces** and it constitutes their common boundary. For the hyperplane with equation $a_1x_1 + a_2x_2 + ... + a_nx_n = b$, the two half-spaces have the following analytic expression:

$$\{x \in \mathbb{R}^n : a_1x_1 + a_2x_2 + ... + a_nx_n \leq b\}$$

and

$$\{x \in \mathbb{R}^n : a_1x_1 + a_2x_2 + ... + a_nx_n \geq b\}.$$

More exactly, these are **closed half-spaces** that contain their boundary.

Definition A.9. A convex polyhedron is an intersection of finitely many closed half-spaces in $\mathbb{R}^n$.

A half-space is obviously convex, and hence an intersection of half-spaces is convex as well. Thus convex polyhedra bear the attribute convex by right.

A half-space is the set of all solutions of a single linear inequality (with at least one nonzero coefficient of some variable x_j). The set of all solutions of a system of finitely many linear inequalities, a.k.a. the set of all feasible solutions of a linear program, is geometrically the intersection of finitely many half-spaces, alias a convex polyhedron. (We should perhaps also mention that a hyperplane is the intersection of two half-spaces, and so the constraints can be both inequalities and equations.)

Let us note that a convex polyhedron can be unbounded, since, for example, a single half-space is also a convex polyhedron. A bounded convex polyhedron, i.e. one that can be placed inside some large enough ball, is called a **convex polytope**.

The dimension of a convex polyhedron $P \subseteq \mathbb{R}^n$ is the smallest dimension of an affine subspace containing P. Equivalently, it is the largest d for which P contains points $\mathbf{x}_0, \mathbf{x}_1, ..., \mathbf{x}_d$ such that the d-tuple of vectors $(\mathbf{x}_1 - \mathbf{x}_0, \mathbf{x}_2 - \mathbf{x}_0, ..., \mathbf{x}_d - \mathbf{x}_0)$ is linearly independent.

Appendix B

The Ellipsoid Algorithm

The ellipsoid method was invented in 1970 by Shor, Judin, and Nemirovski as an algorithm for certain nonlinear optimization problems. In 1979 Leonid Khachyian outlined, in a short note, how linear programs can be solved by this method in provably polynomial time. This was the first algorithm that proved that LP's can be solved in polynomial time. Although the algorithm is theoretically better than the Simplex algorithm, which has an exponential running time in the worst case, it is very slow practically and not competitive with Simplex. Nevertheless, it is a very important theoretical tool for developing polynomial time algorithms for a large class of convex optimization problems, which are much more general than linear programming. More information can be found in [14], [29], or any other book about linear programming.

The ellipsoid algorithm solves a feasibility problem. More precisely, in its initial description it solves the following problem:

Problem 1. *Given a polyhedron $P \subseteq \mathbb{R}^n$, written as $Ax \leq b$, find a point in P.*

We will describe the algorithm for this case, and then we will see how we can adapt it to solve optimization problems. We assume that the constraints are non-degenerate, so that P is either empty or has a non-zero volumed denoted by $\mathrm{vol}(P)$. In other words we can find a lower bound V_l on $\mathrm{vol}(P)$. We start off with an ellipsoid of volume V_u guaranteed to bound P if it is finite. If $\mathrm{vol}(P)$ is infinite, we start with a suitable V_u and we will eventually get to a feasible point anyway. In our case, the initial bounding ellipsoid is a sphere (ball) in $\mathbb{R}^n$. A single step of the algorithm either finds a point in P, in which case we have proved feasibility, or finds another ellipsoid bounding P that has a volume that is substantially smaller than the volume of the previous ellipsoid. We iterate on this new ellipsoid. In the worst case we need to iterate until the volume of the bounding ellipsoid gets below V_l, in which case we can conclude that the system is infeasible.

Before proceeding to a more formal description of the algorithm, we need to define what an ellipsoid is.

Definition B.1. An ellipsoid can be defined as an affine transformation of a ball. We let $B_n = \{x \in \mathbb{R}^n : x^T x \leq 1\}$ be the n-dimensional ball of unit radius centered at 0. Then an n-dimensional ellipsoid is a set of the form

$$E = \{Mx + s : x \in B^n\},$$

where M is a nonsingular $n \times n$ matrix and $s \in \mathbb{R}^n$ is a vector.

An alternative definition is this: Given a center a, and a positive definite matrix A, the ellipsoid $E(a, A)$ is defined as

$$E(a, A) = \{x \in \mathbb{R}^n : (x - a)^T A^{-1}(x - a) \leq 1\}$$

We remind the reader here that a real $n \times n$ matrix A is positive definite if it is symmetric and $x^T Ax > 0$ for all non-zero vectors x. Moreover, a real symmetric matrix A is positive definite iff there exists a real nonsingular (i.e. invertible) matrix M such that $A = MM^T$ (which gives $A^{-1} = (M^{-1})^T M^{-1}$).

The ellipsoid algorithm can be stated now as follows:

Algorithm 8: `The Ellipsoid Algorithm`

```
1. Start with a ball $E_0$ centered at $\mathbf{0}$ and containing the set $P$.
2. At the $i^{th}$-iteration, check whether the center of the current ellipsoid $z_i$ is in $P$ .
    • YES. Output $z_i$ as the feasible point.
    • NO. Find a constraint for $P$, $a_k \cdot x \le b_k$, violated by $z_i$ . Recurse on $E_{i+1}$,
      the minimum volume ellipsoid containing $E_i \cap \{x \mid a_k \cdot x \le a_k \cdot z_i\}$.
```

A pictorial description is given below:

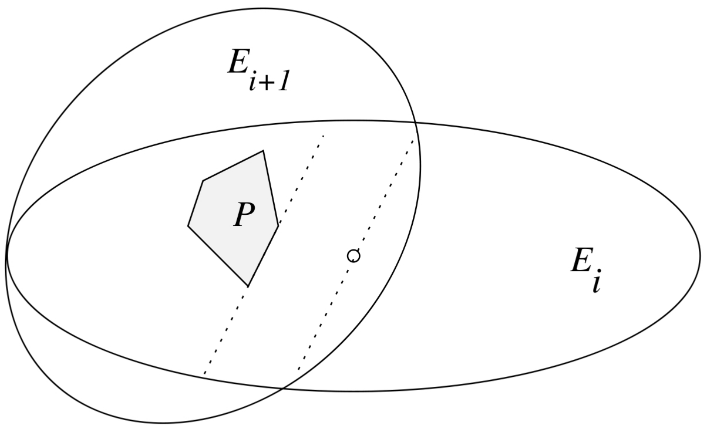

Figure B.1: A singe iteration of the ellipsoid algorithm

Observe that the algorithm halts when a point z_i is found to be within P . It must halt, since at any step i, P is a subset of E_i , and we can prove that after each step, the volume of E_{i+1} has decreased by an appreciable amount. For some value of i, the volume of E_i will be smaller than the volume of P, so the algorithm must halt before reaching this point. We should notice here that we need to deal with the case when P has no volume (i.e. P has just a single point), and also check when we can stop and be sure that either we have a point in P or we know that P is empty. However, we will not go into such details here, and we will only say that there are ways to handle all such cases. The interested reader can refer to the suggested bibliography. Regarding the volume of succeeding ellipsoids, we have:

Lemma B.1. *Let $E \subseteq \mathbb{R}^n$ be an ellipsoid and $H \subseteq \mathbb{R}^n$ be a half-space passing through the center of E. Then there is a unique ellipsoid E' of minimum volume containing the half-ellipsoid $E \cap H$ and $\frac{vol(E')}{vol(E)} \le e^{-1/(2m)}$.*

Thus, for each iteration of the ellipsoid algorithm we get $\frac{\text{vol}(E_{i+1})}{\text{vol}(E_i)} \le e^{-1/(2m)}$. To run the ellipsoid algorithm, we need to be able to decide, given $x \in \mathbb{R}^n$, whether $x \in P$ or find a violated inequality. One of the ways to handle this is by introducing the notion of the separating oracle.

Definition B.2. A polynomial time Separating Oracle for a convex set K is a procedure which, given x, either tells that $x \in K$ or returns a hyperplane separating x from K. The procedure should run in polynomial time.

Note that this idea allows us to use the ellipsoid algorithm in a much more general setting, convex programming, in which the main question is: given a convex set K, find a point x in K. The main thing we need for our algorithm is to be able to answer the question of whether z_i is in P or not and find a separating hyperplane in the latter case. The beauty here is that we do not necessarily need a complete and explicit description of P in terms of linear inequalities. In fact, there are examples in which we can even apply this to exponential-sized descriptions. In any case, it can be shown that for linear programming there always exists such a polynomial time separating oracle.

So, what has only been left to discuss now is how the ellipsoid algorithm can solve LP's. The problem of optimizing an objective function can be reduced to a series of feasibility problems as follows. Consider for example a maximization problem. We start of with an estimate of the maximum value, say c_0 and check for the feasibility of the following system

$$\begin{aligned} c^T x &\geq c_0 \\ Ax &\leq b \\ x &\geq 0 \end{aligned}$$

If the system is infeasible, we know that the optimum is lesser than c_0. We may now decrease c_0, say by a factor of 2 and check for feasibility again. If this is true, we know that the optimum lies in $[c_0/2, c_0)$. This is essentially a binary search to find the optimum with higher accuracies. We get the optimum in a number of steps polynomial in the input size, each step being a call to a feasibility checking algorithm.

Appendix C

Some missing proofs

***Proof of lemma* 4.7.** Let $y \in \mathcal{P}$. Since the polytope $\mathcal{P}$ has $x \geq \mathbf{0}$ as a valid constraint, it follows that $x_S, y_S \geq 0$ for all S. We have that $h(y)-h(x) \geq d\cdot(y-x) = \hat{d}\cdot(y-x)+(d-\hat{d})\cdot(y-x)$. Since $d_S-\hat{d}_S \geq 0$ and $y_S \geq 0$ for all S, we get that $h(y)-h(x) \geq \hat{d}\cdot(y-x)+(d-\hat{d})\cdot y-(d-\hat{d})\cdot x \geq \hat{d}\cdot(y-x)-(d-\hat{d})\cdot x$. Observe that $d_S - \hat{d}_S \leq \omega w_S^{(1)} \Rightarrow (d_S-\hat{d}_S)x_S \leq \omega w_S^{(1)} x_S \Rightarrow (d-\hat{d})\cdot x \leq \omega w^{(1)}\cdot x \Rightarrow -(d-\hat{d})\cdot x \geq -\omega w^{(1)}\cdot x$. Now using the fact that $f(x) \geq 0$ (recall the definition of our objective function h) we get that $-(d-\hat{d})\cdot x \geq -\omega(w^{(1)}\cdot x + f(x)) = -\omega h(x)$. Thus, we have that $h(y)-h(x) \geq \hat{d}\cdot(y-x)-\omega h(x)$, and, so, $\hat{d}$ is an ω-subgradient of $h(.)$ at x.

□

***Proof of lemma* 4.8.** Let $y \in \mathcal{P}$. We have to show that $h(y)-h(x) \geq d\cdot(y-x)$. We know that $h(x) = w^{(1)}\cdot x + \sum_A p_A f_A(x)$, and $f_A(x) = \sum_{e\in U}(1-\sum_{S:e\in S} x_S)z^*_{A,e}$ for every scenario A. Also, observe that $f_A(y) \geq \sum_{e\in U}(1-\sum_{S:e\in S} y_S)z^*_{A,e}$, since $z^*_{A,e}$ is a feasible solution for $f_A(y)$ (no restriction of the dual program depends on the first stage vector). So, we get that $h(y) \geq w^{(1)}\cdot y + \sum_A p_A(\sum_{e\in U}(1-\sum_{S:e\in S} y_S)z^*_{A,e})$. We rewrite the last term as follows:

$$\begin{aligned}
\sum_A p_A\Big(\sum_{e\in U}\big(1-\sum_{S:e\in S} y_S\big)z^*_{A,e}\Big) &= \sum_A p_A\Big(\sum_{e\in U} z^*_{A,e} - \sum_{e\in U}\big(\sum_{S:e\in S} y_S\big)z^*_{A,e}\Big)\\
&= \sum_A p_A \sum_{e\in U} z^*_{A,e} - \sum_A p_A\Big(\sum_{e\in U}\big(\sum_{S:e\in S} y_S z^*_{A,e}\big)\Big)\\
&= \sum_A p_A \sum_{e\in U} z^*_{A,e} - \sum_A p_A\Big(\sum_{S\in\mathcal{S}} y_S\big(\sum_{e\in S} z^*_{A,e}\big)\Big)\\
&= \sum_A p_A \sum_{e\in U} z^*_{A,e} - \sum_{S\in\mathcal{S}} y_S\Big(\sum_A p_A\big(\sum_{e\in S} z^*_{A,e}\big)\Big)
\end{aligned}$$

and therefore we get that

$$\begin{aligned}
h(y) &\geq \sum_{S\in\mathcal{S}} y_S w_S^{(1)} + \sum_A p_A \sum_{e\in U} z^*_{A,e} - \sum_{S\in\mathcal{S}} y_S\Big(\sum_A p_A\big(\sum_{e\in S} z^*_{A,e}\big)\Big)\\
&= \sum_{S\in\mathcal{S}} y_S\Big(w_S^{(1)} - \big(\sum_A p_A \sum_{e\in S} z^*_{A,e}\big)\Big) + \sum_A p_A \sum_{e\in U} z^*_{A,e}
\end{aligned}$$

We can also express $h(x)$ in a similar way replacing y_S with x_S and using equality, and, thus, we get

$$h(x) = \sum_{S\in\mathcal{S}} x_S\Big(w_S^{(1)} - \big(\sum_A p_A \sum_{e\in S} z^*_{A,e}\big)\Big) + \sum_A p_A \sum_{e\in U} z^*_{A,e}$$

Setting $d_S = w_S^{(1)} - \sum_A p_A \sum_{e\in S} z^*_{A,e}$ and subtracting the two terms we get $h(y)-h(x) \geq \sum_{S\in\mathcal{S}}(y_S - y_S)d_S$. Now, to bound $\|d\|$, since $z^*_{A,e} \geq 0$ for all A, e, we get that $d_S \leq w_S^{(1)}$.

Also, since $\sum_{e \in S} z^*_{A,e} \leq w^{(2)}_S$ for all S, we get $d_S \geq w^{(1)}_S - \sum_A p_A w^{(2)}_S = w^{(1)}_S - w^{(2)}_S \sum_A p_A = w^{(1)}_S - w^{(2)}_S \geq w^{(1)}_S - \lambda w^{(1)}_S = (1-\lambda) w^{(1)}_S$. Therefore $|d_S| \leq \lambda w^{(1)}_S$, and hence $\|d\| \leq \lambda \|w^{(1)}\|$. □

***Proof of lemma* 5.5.** We assume that inequalities 5.1 and 5.2 hold with probability at least $1-2\delta$. Let $\bar{x}$ be an α-approximate minimizer for $\hat{f}$ and x^* be a minimizer of $f(.)$. We set $x = \bar{x}$ in 5.1 and $x = x^*$ in 5.2, and we get

$$f(\bar{x}) - \hat{f}(\bar{x}) \leq \varepsilon Z^* + 2\varepsilon c(\bar{x}) + f_h(0) - \hat{f}_h(0)$$

and

$$\hat{f}(x^*) - f(x^*) \leq \varepsilon Z^* + 2\varepsilon c(x^*) + \hat{f}_h(0) - f_h(0)$$

Multiplying the second with α and adding them up together we get

$$f(\bar{x}) + \alpha \hat{f}(x^*) - \hat{f}(\bar{x}) - 2\varepsilon c(\bar{x}) \leq \alpha f(x^*) + (1+\alpha)\varepsilon Z^* + 2\alpha\varepsilon c(x^*) + (\alpha - 1)(\hat{f}_h(0) - f_h(0))$$

and since $c(x) \leq f(x)$, as already mentioned, and $\hat{f}(\bar{x}) \leq \alpha \hat{f}(x^*)$, we get

$$\begin{aligned}
f(\bar{x}) - 2\varepsilon f(\bar{x}) \leq \alpha f(x^*) + (1+\alpha)\varepsilon f(x^*) + 2\alpha\varepsilon f(x^*) + (\alpha-1)(\hat{f}_h(0) - f_h(0)) \Rightarrow \\
(1-2\varepsilon) f(\bar{x}) \leq \alpha f(x^*) + (1+\alpha)\varepsilon f(x^*) + 2\alpha\varepsilon f(x^*) + (\alpha-1)(\hat{f}_h(0) - f_h(0)) \Rightarrow \\
(1-2\varepsilon) f(\bar{x}) \leq (\alpha + \varepsilon + \alpha\varepsilon + 2\alpha\varepsilon) f(x^*) + (\alpha-1)(\hat{f}_h(0) - f_h(0)).
\end{aligned}$$

Using the fact that $\alpha \geq 1$ we finally get

$$(1-2\varepsilon) f(\bar{x}) \leq (1+4\varepsilon)\alpha f(x^*) + (\alpha-1)(\hat{f}_h(0) - f_h(0)).$$

□

Bibliography

[1] B.M. Anthony and A. Gupta. Infrastructure leasing problems.
In *Proceedings of the 12th Conference on Integer Programming and Combinatorial Optimization (IPCO)*, 2007. Also, in Vol. 4513 of *Lecture Notes in Computer Science*, M. Fischetti and D. P. Williamson, Eds., Springer, 424 - 438.

[2] S. Arora and B. Barak. Computational Complexity: A Modern Approach.
Cambridge University Press, 2009.

[3] E. M. L. Beale. On minimizing a convex function subject to linear inequalities.
Journal of the Royal Statistical Society, Series B, 17: 173 - 184; discussion 194 - 205, 1955.

[4] M. Bern and P. Plassmann. The Steiner problem with edge lengths 1 and 2.
Information Processing Letters, 32: 171 - 176, 1989.

[5] J. R. Birge and F.V. Louveaux. Introduction to Stochastic Programming.
Springer-Verlag, NY, 1997.

[6] S. Boyd and L. Vandenberghe. Convex Optimization.
Cambridge University Press, 2004.

[7] M. Charikar, C. Chekuri and M. Pál. Sampling Bounds for Stochastic Optimization.
In *Proceedings of the 9th International Workshop on Randomization and Computation (RANDOM)*, 2005.

[8] V. Chvátal. A greedy heuristic for the set-covering problem.
Mathematics of Operations Research, 4: 233 - 235, 1979.

[9] T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein. Introduction to Algorithms. Second Edition.
MIT Press and McGraw-Hill, 2001.

[10] G. B. Dantzig. Linear programming under uncertainty.
Management Science, 1: 197 - 206, 1955.

[11] K. Dhamdhere, R. Ravi and M. Singh. On two-stage stochastic minimum spanning trees.
In *Proceedings of the 11th Conference on Integer Programming and Combinatorial Optimization (IPCO)*, 2005.

[12] M. E. Dyer and A. M. Frieze. On the complexity of computing the volume of a polyhedron.
SIAM Journal on Computing, 17: 967 - 974, 1988.

[13] M. E. Dyer and L. Stoogie. Computational Complexity of stochastic integer programming problems.
Mathematical Programming, 106(3): 423 - 432, 2006.

[14] M. Grötschel, L. Lovász and A. Schrijver. Geometric Algorithms and Combinatorial Optimization.
Springer-Verlag, New York, 1988.

[15] A. Gupta and A. Kumar. A Constant-Factor Approximation for Stochastic Steiner Forest.
In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing (STOC)*, 2009.

[16] A. Gupta and M. Pál. Stochastic Steiner trees without a root.
In *Proceedings of the 32nd International Colloquium on Automata, Languages and Programming (ICALP)*, 2005.

[17] A. Gupta, M. Pál, R. Ravi and A. Sinha. Boosted sampling: Approximation Algorithms for Stochastic Optimization.
In *Proceedings of the 8th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*, 2005.

[18] A. Gupta, M. Pál, R. Ravi and A. Sinha. What about Wednesday? Approximation Algorithms for Multistage Stochastic Optimization.
In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing (STOC)*, 2004.

[19] A. Gupta, R. Ravi and A. Sinha. An Edge in Time Saves Nine: LP Rounding Approximation Algorithms for Stochastic Network Design.
In *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2004.

[20] W. Hoeffding. Probability inequalities for sums of bounded random variables.
Journal of the American Statistical Association, 58: 13 - 30, 1963.

[21] N. Immorlica, D. Karger, M. Minkoff and V. Mirrokni. On the costs and benefits of procrastination: approximation algorithms for stochastic combinatorial optimization problems.
In *Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2004.

[22] N. Immorlica, M. Mahdian and V. Mirrokni. Limitations of cross-monotonic cost-sharing schemes.
In *Proceedings of the 16th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2005.

[23] K. Jain and V. Vazirani. Applications of approximation algorithms to cooperative games.
In *Proceedings of the 33rd Annual ACM Symposium on the Theory of Computing (STOC)*, 2001.

[24] A. J. Kleywegt, A. Shapiro and T. Homem-De-Mello. The sample average approximation method for stochastic discrete optimization.
SIAM Journal on Optimization, 12: 479 - 502, 2001.

[25] P. Koutris. Infrastructure Leasing Problems.
M.Sc. Thesis, Graduate Program in Logic, Algorithms and Computation, National and Kapodistrian University of Athens, 2010.

[26] R. Levi, M. Pál, R. Roundy and D. B. Shmoys. Approximation algorithms for stochastic inventory control models.
In *Proceedings of the 11th Conference on Integer Programming and Combinatorial Optimization (IPCO)*, 2005.

[27] R. Levi, R. Roundy and D. B. Shmoys. Provably near-optimal sampling-based policies for stochastic inventory control models.
In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing (STOC)*, 2006.

[28] M. Mahdian. Facility location and the analysis of algorithms through factor-revealing programs.
Ph.D. thesis, MIT, Cambridge, MA, 2004.

[29] J. Matoušek and B. Gärtner. Understanding and Using Linear Programming.
Springer, 2007.

[30] R. R. Mettu and C.G. Plaxton. The online median problem.
In *41st Annual Symposium on Foundations of Computer Science (FOCS)*, 2000.

[31] A. Meyerson. The Parking Permit Problem.
In *Proceeding of the 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2005.

[32] R. Motwani and P. Raghavan. Randomized algorithms.
Cambridge University Press, 1995.

[33] C. H. Papadimitriou. Computational Complexity.
Addison-Wesley, 1994.

[34] M. Pál and E. Tardos. Group strategyproof mechanisms via primal-dual algorithms.
In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2003.

[35] R. Ravi and A. Sinha. Hedging uncertainty: approximation algorithms for stochastic optimization problems.
Mathematical Programming, Series A, 108: 97 - 114, 2006.

[36] A. Ruszczynski and A. Shapiro. Stochastic Programming.
Vol 10 of *Handbook in Operations Research and Management Science*, Elsevier, 2003.

[37] A. Shapiro. Montecarlo sampling methods.
Vol 10 of *Handbook in Operations Research and Management Science*, Elsevier, 2003.

[38] D. B. Shmoys and C. Swamy. An Approximation Scheme for Stochastic Linear Programming and its Application to Stochastic Integer Programs.
Journal of the ACM, 53(6): 978 - 1012, 2006. A preliminary version appeared as "Stochastic Optimization is (almost) as Easy as Deterministic Optimization" in *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2004.

[39] A. Srinivasan. Approximation algorithms for stochastic and risk-averse optimization.
In *Proceedings of the 18th annual ACM-SIAM symposium on Discrete algorithms (SODA)*, 2007.

[40] L. Stoogie. Design and analysis of algorithms for stochastic integer programming.
Volume 37 of *CWI Tract. Centrum voor Wiskunde en Informatica*, Amsterdam, 1987.

[41] C. Swamy. Risk-Averse Stochastic Optimization: Probabilistically-Constrained Models and Algorithms for Black-Box Distributions.
In *Proceedings of the 22th annual ACM-SIAM symposium on Discrete algorithms (SODA)*, 2011.

[42] C. Swamy and D. B. Shmoys. The Sample Average Approximation Method for 2-stage Stochastic Optimization.
Unpublished manuscript, 2004.

[43] C. Swamy and D. B. Shmoys. Sampling-based approximation algorithms for multi-stage stochastic optimization.
In *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2005.

[44] C. Swamy and D. B. Shmoys. Algorithms Column: Approximation Algorithms for 2-stage Stochastic Optimization Problems.
SIGACT News, 37: 33 - 46, 2006.

[45] S. Toda. PP is as hard as the polynomial time hierarchy.
SIAM Journal on Computing, 20: 865 - 877, 1991.

[46] L. G. Valiant. The complexity of enumeration and reliability problems.
SIAM Journal on Computing, 8: 410 - 421, 1979.

[47] V. V. Vazirani. Approximation Algorithms.
Springer-Verlag, 2001.