Springer Series in Operations Research
and Financial Engineering

Alan J. King
Stein W. Wallace

Modeling with Stochastic Programming

Second Edition

Springer

Springer Series in Operations Research and Financial Engineering

The Springer Series in Operations Research and Financial Engineering publishes monographs and textbooks on important topics in theory and practice of Operations Research, Management Science, and Financial Engineering. The Series is distinguished by high standards in content and exposition, and special attention to timely or emerging practice in industry, business, and government. Subject areas include:

Linear, integer and non-linear programming including applications; dynamic programming and stochastic control; interior point methods; multi-objective optimization; Supply chain management, including inventory control, logistics, planning and scheduling; Game theory Risk management and risk analysis, including actuarial science and insurance mathematics; Queuing models, point processes, extreme value theory, and heavy-tailed phenomena; Networked systems, including telecommunication, transportation, and many others; Quantitative finance: portfolio modeling, options, and derivative securities; Revenue management and quantitative marketing Innovative statistical applications such as detection and inference in very large and/or high dimensional data streams; Computational economics

Alan J. King • Stein W. Wallace

Modeling with Stochastic Programming

Second Edition

Springer

Alan J. King
Thomas J. Watson Research Center
IBM Research
Yorktown Heights, NY, USA

Stein W. Wallace
Department of Business and Management Science
NHH Norwegian School of Economics
Bergen, Norway

ISSN 1431-8598 ISSN 2197-1773 (electronic)
Springer Series in Operations Research and Financial Engineering
ISBN 978-3-031-54549-8 ISBN 978-3-031-54550-4 (eBook)
https://doi.org/10.1007/978-3-031-54550-4

Mathematics Subject Classification: 90-xx, 90-02, 90-10, 90B15, 90C15

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Paper in this product is recyclable.

To Roger Wets, our mentor and inspiration.

Preface

I do not approve of anything which tampers with natural ignorance.
– *Oscar Wilde*

This book has been in the making since Wallace had a sabbatical in Grenoble, France, in 1996/97, just after Kall and Wallace [47] was published. Already at that time it was clear that the next step would be modeling. Between then and now several intermediate texts have arisen, all being used for teaching over many years, also by others than us. At one point, several attempts to publish the text were made, but failed; possibly because the text was not good enough, certainly because the editors (or their reviewers) did not appreciate that stochastic programming is more than mathematics and algorithms. But maybe that was a good outcome after all. After the two of us teamed up, Wallace with a basic text, King with a publisher and fresh ideas, things have moved forward and all delays have been on us. We don't know how the text will be received, but at least *we* are happy with it. It reflects how we think about our field.

It is now 2024, and it seems like a good time to update the text. Some basics do not change, but of course, our understanding increases. And we felt a strong need to update a specific part of the text, namely the discussion on scenarios, and to add initial thoughts on multistage problems. The reason for the first is that there are no new overviews over scenario generation since the first edition of the book. That section is till rather technical at times, but we assume that only if you are really into the subject will you read beyond (the very important) initial discussions. We have also added a new short chapter on large-scale dependent random variables. That has been added to show that there is hope even in cases with tens of thousands of dependent random variables, and we hope to encourage some of you to look into this area. Multistage models are far from fully understood, but we felt a need to set up an initial frameworks and discuss what are the new questions that need to be asked. We think this material easily can be a start for many theses and papers.

We suppose that our reader faces a decision problem, probably one she is fairly knowledgeable about. But she realizes that uncertainty plays an important role. Very

likely, a deterministic model will not do the trick. What issues need to be faced before she has a finished model?

Passing from a deterministic to a stochastic formulation involves many subtleties. In this book, we try to be very detailed about what the issues are: Why do deterministic models fail to capture stochastics meaningfully? How to think about the information structures? How to rewrite the model so that it becomes consistent with the information structure? How to handle the random variables? How to model risk in order to find a really good solution?

We believe that in many cases stochastic programming provides an appropriate modeling framework, in particular two-stage models, where the first stage is to make some investment and the second to operate this investment under uncertainty. On the other hand, we know that for reasonably large cases, and certainly when we face integer decisions, the resulting models are, for many problem classes, not numerically solvable. However, the reasons for using this framework are so compelling, especially in view of the fact that the deterministic approaches do not deliver what they promise, that we need to find a way through this dilemma.

Real and financial options and insurance contracts provide robust decisions by providing flexibility in the future where uncertainty has to be faced. But all these instruments (decisions) are of the type "Add something on top of something else," for example, adding some capability or capacity on top of the solution to a deterministic program or on top of whatever is being done today. These types of instruments already exist and the main issue is to value them and determine if they are good deals or not.

We show in this book that stochastic programming models provide good solutions because stochastic programming explicitly models the value of future decisions that are made after uncertainty has been revealed. We shall see that the really good solutions, that take optimally into account future decisions, are not of the type "something on top of something else."

So, where does this take us? First, we need to understand *why* deterministic optimization does not bring us where we want to be when modeling uncertainty. Next, we need to be able to formulate what we would like to solve. In some cases, we may even be able formulate and solve exactly what we set out to do; realistic stochastic programs are not always unsolvable. But if not, stochastic programming technologies provide us with tools to explore in what *ways* solutions to stochastic programs are good. Chapter 7 gives a specific example how good solutions come about by explicitly identifying solutions that optimally open up for future decisions.

Understanding why we need stochastic programs, being able to formulate them, and finally, finding out what it is that makes solutions good can help us find good solutions without actually solving the stochastic programs. Therefore, this book is much more than a book on how to build unsolvable models. Rather, it shows a way forward so that we can potentially benefit from a modeling framework that, in our view, is the right one for many decisions facing uncertain consequences.

Related Literature

This text still fills a hole in the present set of available books. On one hand, there are a few rather technical books on how to solve stochastic programs, such as Kall and Wallace [47], Prékopa [73], Birge and Louveaux [9], Kall and Mayer [46], Pflug and Pichler [70], Haneveld, van der Vlerk, and Romeijnders [31], and Royset and Wets [77] and a collection edited by Ruszczyński and Shapiro [78]. On the other hand, a collection edited by Wallace and Ziemba [86] covers a large set of genuine applications and discusses available software at the time. Other applications can be found in the volume edited by Gassmann and Ziemba [28]. Our book picks up what is missing: How to take the difficult step from a verbal problem description to a useful mathematical formulation? We do not discuss efficient solution procedures, as these are available elsewhere.

Audience

The major target groups are graduate students, researchers (academic or industrial), and university professors interested in modeling decision-making under uncertainty. The world is full of uncertainty; what can we do about that when modeling? The book will also be suitable for high-level undergraduates specialized in quantitative modeling.

Required Background

The book assumes the reader already has basic undergraduate knowledge of linear programming and probability, and some introduction to modeling from operations research, management science, industrial engineering, or something similar.

Technical Difficulty

The first part of the book carries all the important ideas and should be read by all readers. This part has a low technical difficulty (apart from some details in Chap. 4), but we intend it to be rather challenging conceptually. This is the reason why we define the audience as above, despite the fact that technically speaking, lower-level students could probably access the book. Chapter 5 is fairly easy to read, and hopefully can serve as an inspiration for many in finding good research questions.

In Chap. 6, there are some rather technical parts toward the end, but the first part should be readable for most readers wanting to start thinking about more than two stages.

The two last chapters contain two examples applying the ideas from the first part of the book. These are not "applications" in the ordinary sense. The focus is: "Here is a problem, how shall we think about it—how shall we model it?" rather than "Here is a successful application of stochastic programming." So, the second part tries to show how we would start thinking about modeling the way we suggest in the first part. In this part the technical is moderate.

Throughout the book we provide pointers to more advanced literature. On every subject we discuss there is much more to be said. However, we have chosen to avoid most details for several reasons. The main one is simply that we wanted to write a book about the *basics* of modeling with stochastic programming. We believe that there is a more serious need for a book explaining the underlying "whys" than the technically deeper "hows." But also, we try to avoid drowning the conceptual difficulties in technical details. We believe that whenever a student or user has the basic questions and difficulties in place, they can always walk down the alleys of detail reflecting their needs and abilities. Another matter is that a book like this, touching upon so many areas related to stochastic programming, and not only discussing stochastic programming in a limited sense, would indeed be quite a brick, if not a collection of bricks. So we prefer to provide a cornerstone rather than material for a whole building.

The Chapters

1. *Uncertainty in optimization.* This chapter introduces all the basic ideas of the book. It contains our main discussion of why sensitivity analysis in deterministic optimization does not deliver what it promises. This is done primarily by a simple example. We then relate our work to real option theory, and present a detailed discussion on the core modeling concepts of robustness and flexibility. We also touch upon robust optimization and chance-constrained models as alternative approaches. The focus throughout is not on describing the mathematics of these approaches, but to discuss what it means to use them. What do the methods imply about our understanding of the world? We also point out that stochastic programming is primarily about transient decision-making. Finally scenario generation gets its first treatment.
2. *Information structure and feasibility.* This is the major theoretical chapter. It builds on Chap. 1. Much of the theory is presented using examples, and rather than focusing on technical intricacies, we again ask, what are the issues, what should be done? Much of the focus is on the issue of feasibility (should we use constraints or penalties) and information structures. We also discuss how to handle tail effects if the problem has too many (often infinitely many) time periods.

3. *Modeling the objective function.* In this chapter, we discuss what to do since a decision leads to an outcomes distribution rather than a number. We discuss when it is appropriate to maximize expected profit, and when risk should be considered. Soft and hard constraints are discussed, and penalty formulations are compared to options modeling.
4. *Scenario tree generation.* When solving stochastic programs, the stochastics must be represented by discrete distributions. This is a side effect of the chosen solution method, and is not caused by the problem under study. This chapter follows up Chap. 1, and explains why a user has to be careful, what can go wrong if one is not careful, and discusses what can be done for those not interested in scenario generation as such, but still need to worry about what they end up doing. We discuss stability questions (relative to the scenario tree generation), statistical approaches to test the quality of solutions (wherever they come from), and we indicate how the two relate. We also discuss in some detail what should be understood by a good scenario tree/discretization and how different approaches differ in this respect. We go into some detail here as scenario generation probably is less settled than all the other issues brought up in the book. At the same time, in our view, doing scenario generation correctly is much more important than what the literature indicates, given that our interest is the model under investigation and not simply the development of an algorithm. This reflects our take on the whole subject of stochastic programming: We are thinking in terms of decision-making and operations research more than mathematical programming as such. This chapter is seriously updated to include what has been done since the first edition of the book was ready.
5. *High-dimensional dependent randomness.* Much of the literature on scenarios are for rather low dimensions, and often (but not always) on independent randomness. Of course, as time goes by, the abilities develop. But as long as we are not willing to pass beyond sampling, things will really never move on. Sampling is not only inefficient in terms of how many scenarios are needed to get good results from the optimization model, but it is also very hard—really impossible in many cases—if the number of variables is very high, they are dependent, and possibly even of different types. The point of this chapter is to show a way out of this difficulty, and using ideas from Chap. 4, we show how problems with tens of thousand of dependent random variables can be solved in a context of vehicle routing. This is not a general approach, but it can show the way for you to consider other problems of similar type. The high dimensions are natural in this context, so we are not discussing problems that are made big just for fun.
6. *Multistage models.* This is a new chapter, and it sets up a framework for how to think about multistage models. We discuss how to handle long tails (in terms of the number of stages in a problem), how to think about information structures, and what new questions arise in terms of scenario generation.
7. *Service network design.* This inherently two-stage example comes from logistics, and is a summary of a thesis doing exactly what this books advocates, step by step. It should be a very useful chapter for a PhD student wanting to embark on a

project of her own. The focus is on modeling, in particular how to create a model which is two-stage from a problem which has infinitely many stages.

8. *A multi-dimensional newsboy problem with substitution.* This chapter discusses an extension of the famous newsboy model, looking at the case with multiple products, dependent demand, and substitution among the products. The model looks at the demand for high-tech fashion products, and it is of particular interest to see how we can model multi-modal demand distributions, which are crucial for this problem.

Yorktown Heights, NY, USA — Alan J. King
Bergen, Norway — Stein W. Wallace
January 17, 2024

Acknowledgments

We gratefully acknowledge the generous assistance of our supporting institutions throughout the time the book has been under construction: IBM Research, NHH Norwegian School of Economics, Lancaster University, the Chinese University of Hong Kong, Molde University College, and The Norwegian University of Science and Technology. Robin Lougee and Hongyan (Jenny) Lu provided essential encouragement and support: the wonderful quotes on uncertainty were collected by Robin, and Jenny provided non-stop cooking to fuel the final sprint. For this, and for so much more, our heartfelt thanks.

Contents

Chapter 1
Uncertainty in Optimization

Life is uncertain. Eat dessert first
– *Ernestine Ulmer*

Decisions are rarely made under certainty. There is almost always something relevant about the future that is not known when important decisions are made.

- What will the weather be during the outdoor concert?
- Will my technology capture the market, or will my competitor capture all the sales?
- Will the government change the taxation rules?
- Will the killer asteroid hit during lunch?
- Will my salmon farm be infested with lice?
- How cold will it be this winter (and hence what will be the energy demand)?
- What will demand be for my product the next ten years (as I am making a major investment in production equipment)?
- Will the Chinese currency become convertible?
- What is the travel time to my outlet in central London?

There is no end to what we do not know. Still, we must make decisions since the problems are there and need attention. The key central questions to modeling uncertainty in decision problems will always be:

- What are the important uncertainties?
- How can we handle them?
- Can we deliver valuable solutions and insights?

Stochastic programming is a part of mathematical programming and operations research that studies how to incorporate uncertainty into decision problems. The canonical expression of the decision problems we seek to handle in stochastic programming goes as follows:

A. J. King, S. W. Wallace, *Modeling with Stochastic Programming*, Springer Series in Operations Research and Financial Engineering,
https://doi.org/10.1007/978-3-031-54550-4_1

Some decisions must be made today, but important information will not be available until after the decision is made.

Problems of this type are found in transportation, logistics, supply chain management, capacity planning, financial asset management, and other related fields. The problems show up in government as well as private enterprises. Stochastic programming provides modeling and solution techniques that harness the power of optimization to solve models that are sensitive to the impact of uncertainty.

This book is about modeling decision problems in order to obtain solutions that perform well under uncertainty.

This book is also about what the parts of this statement mean: about what a "solution" is and what it means for solutions to "perform well" in applications and environments with much uncertainty.

Before embarking upon this text you have most likely had a modeling course in operations research, management science, mathematical programming, or industrial engineering. It is likely that only a minor part of the course covered uncertainty in decision-making. Maybe there was a section on sensitivity analysis in optimization, one on stochastic dynamic programming, one on decision-trees, and one on queuing. Did you notice that sensitivity analysis in optimization does not connect very well to the setups for modeling uncertainty? Possibly you did, but the course did not bridge that gap. This book explains what to do next and how to make the connection between optimization and uncertainty.

Current and future operations research professionals, such as yourself, should learn to analyze uncertainty from the perspective of stochastic programming. Thinking about how to model in an uncertain, dynamic world could be one of your most important contributions to decision-making.

1.1 Sensitivity Analysis, Scenarios, What-If's and Stress Tests

Most, if not all, operations research textbooks advise us to be aware that when we formulate a mathematical programming problem some of the numbers we need, such as demand, cost, and quality, are not really known to us. This could be either because they are truly unavailable (e.g., the demand for sugar next year) or it is a question of ignorance (e.g., the sales of sugar of different types in Namibia last year). The common practice in such cases is

- Use statistics, experts, whatever you have to learn about the parameters.
- Solve your model with expected values put in everywhere.
- Observe that your optimal solution $\hat{x}$ depends on the parameters you used.
- Apply *sensitivity analysis* to determine whether $\hat{x}$ depends critically on the parameters.

Sensitivity analysis evaluates how much parameter values can vary without changing the fundamental character of the solution. The general idea is that if you can vary the parameters a lot, your solution is stable relative to variations in the parameters, while if you can only vary them a little, you have an unstable solution. The latter case is cause for concern, since the parameter estimates may be off and hence you may easily be in trouble, while in the first case an error in your estimation is not too dangerous. By using the tool *parametric optimization*, you can see how the solution (and the dual variables in the case of linear programs) change as you move away from your initial values. You can then obtain a good feeling for which parameters are critical, and where you should put in extra efforts to obtain better estimates. By this type of analysis, most textbooks claim, you can determine if uncertainty is important or not.

What-if analysis, *scenario analysis*, and *stress test* approaches all generate a set of possible futures and ask "what if each of these turn out to be the case?". What-if analysis is heavily used in problems involving discrete/integer variables and addresses questions like "what if we open a facility in Hong Kong?" or "what if our competitor opens a distribution center in China?". Scenario analysis and stress tests (in our view) are better applied in situations where there is a need to explore different trend lines in fundamental parameters, such as employment growth, interest rates, or commodity indexes.

All these techniques represent exactly the same kind of approach: to explore a variety of future states of the world and think through the consequences for each one in isolation. Sensitivity analysis is really equivalent in spirit to these what-if's, scenarios, and stress tests: all are used to generate a set of possible optimal solutions for the next step of the decision process. For most of us, this thinking is very intuitive.

A deeper look suggests that sensitivity analysis approaches are likely to miss important features of the solution. They cannot propose any kind of solution that addresses variability. They cannot model the natural process of investing in technologies that are useful in responding to uncertain outcomes.

> Decisions made on the basis of sensitivity analysis and its relatives can only find solutions that are optimal for some fixed (i.e., deterministic) setting of the uncertain parameters.

These approaches are so common that we feel a need to discuss them very early in this book. Our wish is that you develop a good understanding of what a combination

of a deterministic model with expected values, followed by sensitivity analysis, gives and what it does not give. Our own experience is that while many students find it easy to accept that stochastic programming is an appropriate modeling framework when facing uncertainty in a model, it is more difficult to grasp why *sensitivity analysis normally is not a good framework for approaching uncertainty*.

1.2 The NewsMix Example

This section contains an example to illustrate important aspects what can go wrong if sensitivity analysis combined with parametric optimization is applied to a decision problem facing uncertainty, followed by a discussion of how to avoid these difficulties.

The example models a variation of the classical newsvendor's problem, who must make a standing order of newspapers that will be delivered at the start of every day for some period of time:

- There is a maximum limit of 1000 on the newspaper order, caused by the size of the delivery truck.
- The newsvendor can choose between three papers: a political newspaper, a business newspaper, and a regional newspaper.
- The profit from selling one political newspaper is \$1.30, from one business newspaper \$1.20, and the regional newspapers yield \$1.00 profit.
- Survey data shows the first choice of customers is either the political or the business paper, but their second choice is almost always the regional paper.

We make the modeling assumption that the demand for political and business papers sum to exactly 1000, though we know that it is not totally true. Let D_p be the demand for the political newspaper. The demand for the business newspaper is then almost certainly $1000 - D_p$. Table 1.1 summarizes the data for the problem.

We start by formulating the newsmix choices as a deterministic optimization model. For convenience, we normalize the numbers by dividing through by the maximum order. Orders are ratios of the maximum order (so they are between 0 and 1) and D_p is the ratio of demand for political newspapers.

Table 1.1 Data for the newsmix example. The ∞ used in the demand for regional newspapers simply represent that we can sell whatever we order within our capacity

Newspaper	Political	Business	Regional
Quantity ordered	x_p	x_b	x_r
Quantity demanded	D_p	$1000 - D_p$	∞
Profit	\$1.30	\$1.20	\$1.00

$$\max_{x_p,x_b,x_r} \quad 1.30x_p + 1.20x_b + 1.00x_r$$

$$\text{such that} \quad \begin{cases} x_p & & & \leq D_p \\ & x_b & & \leq 1 - D_p \\ x_p & + x_b & + x_r & \leq 1 \\ x_p, & x_b, & x_r & \geq 0 \end{cases} \tag{1.1}$$

In model (1.1) the newsvendor maximizes the "profit" (objective function) over all orders $[x_p, x_b, x_r]$ that do not exceed demand, sum to no more than one, and are non-negative.

1.2.1 Sensitivity Analysis

Since the ratio D_p is not known, a reasonable initial approach may be to solve model (1.1) for *any* possible value of D_p. The optimal solutions are in this case easy to find and are given by

$$\begin{aligned} x_p &= D_p, \\ x_b &= 1 - D_p, \\ x_r &= 0 \end{aligned} \tag{1.2}$$

Suppose now that the forecast for the ratio of political to business newspaper demand is $\bar{D}_p$. Then by (1.2), we can find the optimal order quantities for this estimated demand, namely

$$\begin{aligned} \bar{x}_p &= \bar{D}_p, \\ \bar{x}_b &= 1 - \bar{D}_p, \\ \bar{x}_r &= 0 \end{aligned}$$

We will put this solution into the newsmix model (1.1) and see what happens if we one day observe a demand ratio $\hat{D}_p$. The constraints become:

$$\begin{array}{lll} \bar{D}_p & & \leq \hat{D}_p \\ & 1 - \bar{D}_p & \leq 1 - \hat{D}_p \\ \bar{D}_p + 1 - \bar{D}_p + 0 & & \leq 1 \end{array}$$

Look at the first two. One of them must be violated for any $\hat{D}_p$ that is not precisely equal to $\bar{D}_p$. Unless the demand turns out to be *exactly* $\hat{D}_p = \bar{D}_p$, the model will be infeasible!

What possible conclusion can we draw when our model is practically always infeasible? Does "infeasibility" even make sense in this situation? Well, it probably does not. The fact is that the model we started with is just badly suited for analyzing the problem for unanticipated data values.

An often overlooked step in modeling is to determine how to handle *unanticipated values* of the model data.

You may think this is obvious. Surely you can think of several better models than (1.1) for the newsvendor. But what if the problem was really, really complex with lots and lots of parameters and constraints. Would you be able to tell that the problem was badly formulated? Parameter estimates are *always wrong*. They are just estimates, after all. The difficulties we just observed in our little problem may be hidden within a large model, and it will not be possible to directly observe what is going on.

Do not trust that you can *see* whether or not a large model is suited for parametric analysis. You need to prepare for it by proper modelling.

When the modeling does not tell you what to do when parameters turn out to be wrong, then it is likely that serious issues will emerge only when investments have been made and plans put into practice. Deterministic models, by definition, do not say anything about what to do when parameters are not as expected. For stochastic programs, on the other hand, modeling what might happen and how to handle each and every situation is the core of the modeling.

The property we have just observed is called a *knife-edge*. Optimization procedures tend to do anything to save even a cent, and this results in solutions that fit perfectly for the given parameters but can be very bad for even very similar values. They balance on a knife-edge. We have just modeled an extreme case of this for the newsvendor, namely a solution that is optimal for one parameter value, but infeasible for all others! This was to a large extent caused by weak modeling. But even in more well behaved cases, we may observe these knife-edge properties. To some extent this property is the major reason why solutions from deterministic models can have very bad average behavior.

1.2.2 Information Stages and Event Trees

The difficulty we faced when analyzing the newsmix formulation could have been addressed if we had equipped the model (1.1) with *stages*.

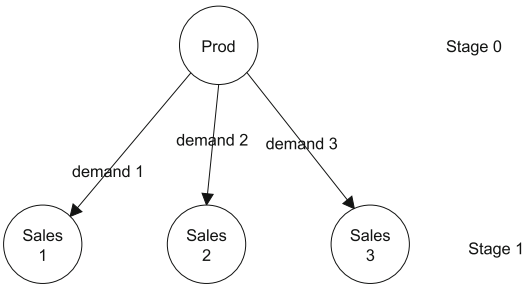


Fig. 1.1 Event (or scenario) tree describing a stage structure for the newsmix example

An information stage (normally just called "stage") is the most important new concept that distinguishes stochastic programming. A stage is a point in time where decisions are made within the model. Stages sometimes follow naturally from the problem setting, and sometimes are modeling approximations.

> It only makes sense to distinguish two points in time as different stages if we observe something relevant inbetween.

"Observing something relevant" must be widely understood. If we hope to receive some data before Tuesday morning but it did not arrive, then that is, of course, information: we now know the data did not arrive, and so we must do without it.

The importance of stages was not so easy to see in the deterministic version of the newsmix model (1.1). But when we implemented the particular solution $\bar{x}$ and saw what happened when demands became known on any given day, then it became clear that there are in fact *two stages*, as illustrated by the event (or scenario) tree in Fig. 1.1.

In *stage 0*, orders are placed. In *stage 1*, after demand for newspapers has become known, we determine what can be sold. With the two stages in place, it seems rather clear that letting the same variables represent both orders and sales is not a very good idea. Because that is what happened: the variables were interpreted as order ratios, but some of the constraints referred to sales. In a deterministic setting, you will never order something that cannot be sold so using the same variables for orders and sales makes some kind of sense.

> In a deterministic world you will never buy or produce something you will never need. This often leads to models with structures that do not even allow meaningful sensitivity analysis.

We have already noted that (1.1) is reasonably well defined in a deterministic environment. We have also noted that it produces rather strange conclusions when it is analyzed in light of the uncertain demand, and that one reason for that is that the stage structure is not properly reflected in the formulation.

Let us review the situation. We will receive demand for political and business newspapers that is close to 100% of our maximal total order, but at order time, the split between political and business newspapers is unknown.

What happens if we choose one of the solutions from the deterministic model (1.1), say $\bar{x}_p = \bar{D}_p$, $\bar{x}_b = 1 - \bar{D}_p$, and $\bar{x}_r = 0$, but it turns out that the actual ratio does not equal the estimate $\bar{D}_p$? Will the world end? Of course not. Will the newsvendor go broke? Probably not. Will nothing be sold since we got it wrong? Unlikely. But the interpretation, within the model, of something being "infeasible" is exactly that: cannot be allowed to happen under any circumstance.

If we get the estimates wrong, it is more likely that we face a penalty, either directly or indirectly in terms of reduced sales. Now, what happens if we order the wrong amounts (which we almost certainly do)? The fact that the order will be wrong must be incorporated in our model for it to be useful. As you have noted, the stochastic setting with stages has already forced us to ask a lot of questions that are not needed in a deterministic setting, and there are more to come.

1.2.3 A Two-Stage Formulation

Let us think about our model in two stages. First we order x_p, x_b, and x_r, constrained by a sum of one. Then demand becomes known, that is, the customers show up, and we learn the values of demand ratios for political papers D_p and business papers $1 - D_p$ (remember that we have assumed that these ratios sum to one). How much can we now sell? Let y_p be the ratio of papers sold as political newspapers and y_b the corresponding ratio of business newspapers. In the second stage we have the following constraints on y.

$$y_p \leq \min\{x_p, D_p\}$$
$$y_b \leq \min\{x_b, 1 - D_p\}$$

When our customers cannot find their preferred newspaper, they will choose the regional paper. The second-stage constraints on the regional paper are

$$y_r \leq \min\{x_r, \max\{0, D_p - x_p\} + \max\{0, 1 - D_p - x_b\}\}.$$

This constraint says that if we do not order any regional papers (that is, $x_r = 0$), then we cannot sell any regional papers. But of course, we know that we will order the wrong amounts of political and business papers. So one of the terms in the max expression will be positive. So either we will not have enough political papers

($D_p - x_p > 0$) or we will not have enough business papers ($1 - D_p - x_b > 0$). Either way there will be a demand for regional papers!

Thinking in two stages has revealed an important aspect of the newsmix problem, namely that the demand for regional newspapers is always going to be non-zero, caused by customers not finding their first choice paper. This is something we see when we analyze this problem in terms of stages. Now, perhaps a good business person does not need a course in stochastic programming to see this point. They would always order a few copies of the regional just because it is not good business practice to turn away customers!

But do you see that this is somewhat strange? Under *no* demand scenario do people really want the regional paper. They always prefer either the political or the business paper. But the newsvendor cannot know in advance what this demand will be, and so in order not to turn away too many customer the vendor will order some regional papers. In a sense, the newsvendor is buying a hedge against the uncertain demand for political and business papers. This is worth an observation:

> Even if some property is present in the optimal solution to *every* deterministic version of a problem, this property does not necessarily carry over to the optimal solution when uncertainty is taken into account.

In fact, the chance that such a property carries over is rather slim. This is crucial to observe: There are aspects of a good solution that deterministic models will never reveal, irrespective of how much you analyze them.

1.2.4 Thinking About Stages

A useful model needs to capture the salient features of the decision problem. The analysis required to model the sequence of interrelated decisions includes much more than just estimating distributions in situations where the deterministic model would require numbers. New questions arose. We had to *identify stages* and *define variables consistent with the stage structures*. We believe that a good modeller should think carefully about stages independently of what kind of model is ultimately used. The stochastic thinking simply helps in understanding the decision problem better or, at least, helps to shed light on how little we have understood.

Questions related to feasibility showed up in our example by questions like "What if we have ordered too little?", "What if we have political newspapers left, but unsatisfied demand for business newspapers?". We also note that there are numerous stochastic versions of one deterministic model. For many of the questions we asked—and gave answers—there are other potentially correct answers, and they would have led to different stochastic models.

A useful approach is always to consider the time-sequence in which decisions are made, and the amount of information available at each point in time. If this exercise identifies important relationships between the decisions, it may be necessary to incorporate stages in the model. This is true whether you end up with a stochastic or a deterministic model.

1.3 Deterministic Models and Options

What-if analysis and parametric optimization imply the solution of many instances of a deterministic model. Using these tools means that we realize that the world is stochastic, but not that we need to make a decision without knowing which of the scenarios will materialize. We look at them one at a time. We could ask, would you buy insurance on your house (if you have one)? Well, you would say, probably I would since otherwise the loss if it burns will be too much for me to handle. The probability of a fire is low, but even so ...

But in a what-if setting, there would be two questions:

- Would you buy insurance on your house if you knew your house would not burn?
- Would you buy insurance on your house if you knew it would burn?

The answers would probably be no in the first case (but many would like to sell it to you) and yes in the second (but who would sell it to you?). Not very useful, right? One yes and one no, and we already knew those were the two possible decisions. And the whole logic is kind of nonsense, right? Insurance makes no sense in a deterministic world. Insurance against what?

Insurance is nothing but the simplest form of an option. You pay a price (the cost of the insurance), get nothing if your house does not burn, but the total value of the house if it burns. So in a sense, whatever happens you have (the value of) the house, but you have paid a price for that certainty—the insurance premium.

So options make no sense in a deterministic world. Options are bought (or sold for that matter) "just in case something happens." But nothing happens in a deterministic world. Having many deterministic worlds does not change that. The worlds you look at are different, but all are equally uneventful.

In a somewhat different context, assume you are running a transportation company, and you might need an extra truck next week. You have three possible approaches

1. Agree now to rent a truck next week from a rental company at a price p_0.
2. Pay a reservation fee r to the rental company now to have the right (but not duty) to rent a truck next week at price p_1.
3. Do neither and hope for the best (or take a serious cost next week when you lack a truck).

Most likely $p_0 < r + p_1$, while $p_0 > p_1$. In this case it seems reasonable (right?) that there exists a probability for you needing the truck such that paying

the fee (buying the option) is optimal. But if you run this in what-if mode, you will either rent at p_0 or not rent at all. The option will never be optimal. And this shows what you need to understand about why scenario analysis/what-if analysis does not deliver:

> Solutions from deterministic models will always be option free. And this cannot be overcome by solving many deterministic models. Hence, solutions based on scenario analysis will always be option free and inflexible.

1.4 Appropriate Use of What-If Analysis

There is one case where what-if analysis will yield the correct answer. And that is when the full first-stage decision is the same for all values of the random variables. If what we do under full information is not changed by that information, the suggested solution will *always be optimal in hindsight*. In this situation flexibility with respect to the future is unnecessary.

> If what-if analysis (or parametric optimization) leads to the same full first-stage decision for all possible values of the random variables then this is indeed the optimal solution also to the corresponding stochastic programming model.

It may be useful to consider a case, for example, where the profit of an operation is dependent on a random variable, but the optimal decision is not. Suppose the random variable is demand. If demand is high, you get rich, if demand is low you barely get by—but whatever happens, the optimal choice is always to build a plan of a given fixed size. The stochasticity is important but you cannot do anything about it. In such a case, a deterministic model will give the same solution as a stochastic model.

Do not confuse this case with the situation where a *certain aspect* of the solution is present in all optimal solutions of *deterministic* versions of a problem. In the newsmix problem, the optimal solution never contained regional newspapers, despite the fact that it is clearly optimal to order some of them. It is only when the *whole* solution is unaffected by uncertainty that you can draw the conclusion that uncertainty does not matter.

If what-if analysis (or parametric optimization) leads to a certain aspect of the solution being the same for all possible values of the random variables, there is no reason to conclude that this aspect will also be part of the optimal solution to the corresponding stochastic programming model.

If a certain aspect of a solution shows up in all what-if solutions, you may be facing something that needs to be done in any case (like establishing a certain flight for an airline since without it the whole network would make no sense) or you may see the effect of ignoring stages (like ordering no regional newspapers in our newsvendor model). Distinguishing one from the other may not be easy, but trying to do so might be very fruitful.

Deterministic Decision-Making

Apart for the special situation just discussed, the general observation is as follows:

Sensitivity analysis is an appropriate tool for analyzing *deterministic* decision problems.

Maybe it is not so surprising that a deterministic tool functions best for deterministic models. The additional complexity introduced by uncertainty in stochastic models makes them more difficult to analyze. New techniques are needed. Simple tools are often used out of expedience, but the usefulness of the results is lacking. There are better alternatives. Using simple tools will not make the complexity of a model go away.

So sensitivity analysis will work in the following situation. Assume you will be making a decision next year. When the time comes, everything will be known. But presently many parameters of the problem are uncertain. You ask yourself: What will I be deciding? If you need to create a budget for next year, what are reasonable estimates for costs or profits? This is a deterministic problem analyzed under uncertainty, and a deterministic model combined with sensitivity analysis will be an appropriate tool. The reason is that no decisions are made under uncertainty. The problem has only one stage, and hence a model with one stage (a deterministic model) will be the right choice.

Note that we sometimes use what-if questions to represent decisions: “What if we buy our competitor?”, “What if we outsource our warehouses?”. This has nothing to do with uncertainty. This is simply a way of expressing a possible decision, and there is nothing wrong in stating a problem in such a fashion. But if we ask: “What if our competitor outsources her warehouses,” because this would seriously affect our business, and we are uncertain as to what she will do, then using a “what-if” approach (as discussed in this chapter) is not wise. We need to take account of the

uncertainty and come up with a decision in light of the uncertainty and the need be able to face both outcomes. It is important to see the difference between these two cases.

In engineering we often find deterministic models which are of the worst-case type. A deterministic model is set up to make sure that a bridge, for example, can take a load of at least a certain weight. We should not mistake this deterministic model for a deterministic problem. The bridge most likely can support a load different from the one specified (probably higher), but there is also a chance that it will be subjected to forces not previously observed. Winds of unusual strength and direction not previously seen may occur, or the bridge may be hit by a ship, or there may be a fire that affects the load-bearing properties of the steel. Statements about the performance under future loads will always be probabilistic. Deterministic modeling does not change the stochastic nature of a problem, and statements of the type . . . will (always) . . . are (almost) never true.

1.5 Bad But Useful

Having spent quite a bit of time on why deterministic models using what-if analysis do not deliver, and even explained what what-if analysis does, we need to ask: Are deterministic models really totally useless? Or at least, if your hunch that stochastics matter in a certain context is correct, can't you learn anything from deterministic models?

If you simply assume that the deterministic model will be useful, you could be in serious trouble. We hope we have explained well why that is the case. Is there nothing to add? Let us pick up our observation from Sect. 1.3 that deterministic solutions are option free. So, to some extent, we know what is wrong with the solutions from the what-if and scenario analysis; they lack investments in flexibility and they lack options, respectively.

Think about network design for a moment. Here you are to set up a network (for example a water distribution network) in terms of structure (where to put pipelines) and capacity of the individual links. You know where the sources and demand points are, and let us assume that demand is random. If you solve a network design model with expected demand, it is fairly obvious that you would end up with a network where you would lack capacity for almost 50% of the scenarios. And that is simply too much. A fist observation is therefore:

> If you are to solve a deterministic model where some decision variables concern capacity, it is probably wise to use a demand/need well above the mean.

This was discussed in [87]. In that specific context it was also found that the network structure from the deterministic models was very good, often optimal. This is of course not a general result, so numerical testing was needed to check this out. But the general idea remains: It could be wise to check (by solving smaller cases of both the stochastic and deterministic version of the problem) if some variables are actually set well by the deterministic model. If that is the case, as it was in the cited paper, a good heuristic for the stochastic model is to first solve the deterministic model to obtain the network structure. These variables are then fixed in the stochastic model—which then turns into a stochastic linear program for the network design model—and capacities are set there. Something for your problem?

> Even if the deterministic solution is bad, it may contain useful information. But that must be checked numerically.

It is not always that this splits into discrete for the deterministic model and continuous for the stochastic one (though that is wonderful when it happens). It could also be some other split. But it is always worth checking.

1.6 Robustness and Flexibility

> I can't change the direction of the wind, but I can adjust my sails to always reach my destination.
> – *Jimmy Dean*

When making decisions under uncertainty, we may look for decisions that (1) help us withstand random events or (2) help us accommodate the events. We define these decisions as robust and flexible, respectively. To picture the situation, a tree is robust with respect to heavy winds, while a grass is flexible. Some define flexibility as the ability to bring a system from a disturbed state back to where it "should" be. A close look will show that this definition is the same as ours.

1.6.1 Robust or Flexible—A Modeling Choice

A decision does not happen to be robust or flexible. These are qualitative properties we force upon them via a model. In any solution to a model, there will be some decision variables that are robust relative to a random phenomenon, others that are flexible. If there are several random phenomena, as there normally are, a variable may be robust relative to one of them, and flexible relative to another. The choice

simply depends on whether the decision is made before or after the realization of the corresponding random variables.

To illustrate the robust versus flexible issues, let us introduce a pair of what we will come to call "inherently two-stage" problems a bit further on in the book.

- *Truck Routing.* You are setting up routes for the trucks of a trucking company. It is part of the agreement with the labor unions that the routes will be kept unchanged for six months. But the demand is both varying and uncertain. How do you set up the routes for the next six months, when you take into account that you will be able to send the goods with any of your trucks as long as the goods get to their destinations on time.
  If we set up a two-stage model, where the first stage determines routes for a the trucks (in light of random demand) and the second stage, after demand has become known, determines how to dispatch the goods on the trucks, we have *modeled* a situation where truck routes are robust and dispatching is flexible relative to the random demand. If we also allow partial rerouting of trucks (at a cost) in the second stage, then truck routes are also (partly) flexible. This example illustrates a general principle: In a two-stage model, robustness is in the first stage (truck routes) and flexibility (dispatching) is in the second stage. Notice that the terms flexible and robust are purely qualitative. Being robust does not mean being good or bad, it just means that uncertainty is met by doing the same whatever happens. The trucks have schedules which they follow irrespective of realized demand: The truck schedules are robust. Since we allow dispatching to use knowledge of demand, dispatching is flexible with respect to random demand.
- *Sports event.* You regularly organize a large sports event. But you are troubled by bad weather. If it rains few people show up. How can you protect yourself against the negative effects of the weather?
  Let us assume you want to maximize expected profit. What can you do? There are some first-stage decisions you can make. One might be able to build a roof on your stadium. The roof obviously provides robustness relative to the random weather. You may also pay an insurance premium against bad weather. The insurance provides robustness against bad weather, not by making the weather irrelevant for the sports event, but by providing flexibility in how you obtain income: Instead of running the event you can collect on the insurance. You may also be able to pay a fee for the right to use another stadium in the case of bad weather. The right provides robustness against bad weather by providing flexibility in the operational phase: You either use your own stadium or, in the case of bad weather, the alternative one.

The quality of a solution is defined relative to the objective function. But robustness and flexibility are defined relative to how the actions represented by the variables are set in time as compared to the realizations of the random variables. This is not how all people use the terms. Many use "robust" to mean a solution where the objective function is stable (does not vary too much) over the realizations of the random variable, e.g., "This plan is robust as my income is almost the same

whatever happens. I am protected against the uncertainty." Some will use only variables that are robust (in our terminology) to achieve this stability (robustness) in income; others will also use flexibility, depending on their modeling philosophy. Our view is that robustness and stability are qualitative properties of decisions and represent different strategies to meet uncertainty. So we reserve the terms "good" and "better" to refer to properties of the objective function.

If stability is required, it will be expressed in the objective function or as a constraint, in which case "good" will be a property of decisions that give stability. The reason for this is maybe a bit subtle: While there is always a need to face the uncertainty, stability in the objective function is often not a goal. So we will rather make statements like: "These truck schedules are robust relative to the random demand, the routing of goods is flexible, and the overall expected profit is maximized." Or: "The truck schedules are robust, but with some minor flexibility in changing them after demand is realized, routing of goods is flexible, and as required, the income stream is stable." We do not want "robust" to be reserved for the special case of income stability, which often is not required.

Assume you have some initial solution (coming from a model or simply being the normal way of doing business). We may then ask: Are there any (real) options available for which the initial deterministic cost (the option cost) is smaller than the expected future value (the option value)? If the answer is yes, you buy the option, if not, you do not. Notice that buying an option is a robust way of facing uncertainty, as it takes place before you observe the outcome of the random variable. The option provides flexibility in a later stage. This connection between robust and flexible is typical and is best seen in the light of options:

> Buying (or selling) an option is a robust way of facing uncertainty, and it has value because it increases flexibility in a later stage.

Robust or Flexible?

In some cases, we wish to have robustness. In other cases we want flexibility. In some cases, either one will do. A bus schedule, for example, should be robust, as passengers normally will hate repeated changes in the schedule. The published schedules (i.e., the services offered) in a company like DHL or FedEx should also be robust, while we will not mind if the routing decisions are flexible, meaning that goods can be sent along many different routes. In a production process we might find flexibility and robustness equally interesting if the resulting expected costs are the same.

Although far from being the main reason for choosing between a robust and a flexible approach to handle a random phenomenon, you should be aware that there is a connection between this choice and the question of bounding a decision problem under uncertainty. This way of thinking can sometimes be very useful. Let us illustrate this by two examples.

Flexibility Bound
Options pricing in mathematical finance assumes that the instrument can be traded all the times, that is, continuously *without any transaction costs*. So this is a perfectly flexible decision setting. This is an optimistic view. For a seller, this will be an upper bound; for a buyer, a lower bound. If you choose to model something more flexible than it really is you obtain a bound. Sometimes this is a wise thing to do, not because you want a bound, but because the actual situation is simply too complicated. One nice effect of thinking like this is that you may be able to verify that you have a bound rather than just an approximation.

Robustness Bound
In a robust approach to decision-making models have only one stage. There is no modeling of what happens after we have learned what happens. If there exist ways to react to uncertainty after uncertainty has been revealed, then the robust approach will be pessimistic. For example, if you are setting up truck schedules for a transportation company, and you assume that whatever happens, you cannot change the schedule, then you have made all scheduling variables robust. If in reality you could change the schedule to some extent after you know what happens, then your robust approach is a bound.

You can use the choice between robustness and flexibility to bound a problem.

1.7 Transient Versus Steady State Modeling

In modeling, we often distinguish between transient and steady state models. A steady state model is one which, for all states we can end up in, tells us what to do. There might be states that are only reachable if we do not make optimal decisions. These are often excluded in the calculations. Transient models, on the other hand, start out with observing where we are, and tell us what to do now. That is, we are somewhere (possibly a very bad place, possibly a perfect place), we have a goal, some resources and constraints, and we try to do as well as we can.

Stochastic programs are primarily about transient decision-making.

The starting point of a stochastic program is the present situation. The model is intended to tell us what to do in light of our goals, constraints, and resources. We might, for example, be running a transportation company. We have a fleet, some customers must obey some rules, and we have a goal for our company. We then ask

what to do. We are not asking what to do in all possible situations, just what to do based on our present state.

If there is no memory in a model then there is no principal difference between steady state and transient modeling. But if there is memory, which is normally the case for important decisions, stochastic programming is primarily about transient modeling. We are where we are, be that good or bad, and we want to know what to do. Even optimal decisions might leave us worse off than we were before we made the decisions if constraints are tough and resources limited.

1.7.1 Inherently Two-Stage (Invest-and-Use) Models

An inherently two-stage model is a model where the first decision is a major long-term decision (or investment), while the remaining stages (or stage) represent the use of this investment. This can be such as building a factory for later production under uncertain demand, prices, or even products. Or it can be to set schedules for a certain period of time for an airline, schedules that will face uncertain demand and flying conditions as well as technical problems on planes. In these cases, mathematically speaking, the first stage will look totally different from the remaining stages, which all will look more or less the same (barring some differences caused by seasons, day-of-week, and the like). This is a very important class of models, possibly the most important one for stochastic programming.

1.7.2 Inherently Multi-Stage (Operational) Models

In inherently multistage models all stages are of the same type. Typical models come from financial trading, production/inventory modeling, or some vehicle routing models. Also here we face such as random demand, prices, or even products, but the models are not split into an investment part and a usage part.

We should not confuse information stages with time periods. Stages model the flow of information and describe the points in time where decisions are made based on the arrival of new information. Time periods, on the other hand, represent the tick of the clock in a model.

Electricity Production Model
To illustrate the basic issues in modeling stages and time periods we introduce the problem of electricity production.

- *Electricity production.* You are responsible for production planning for a major utility company. You face uncertain demand, uncertain rainfall for your hydro-based production and uncertain wind for your wind-mills. How do you plan your operations for the next day?

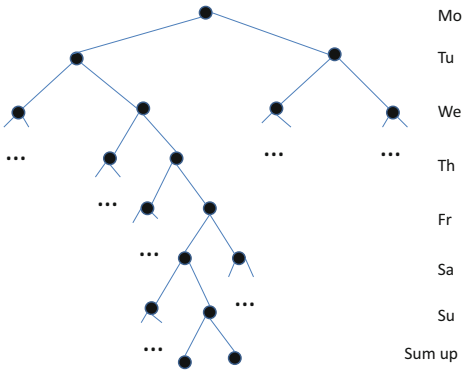


Fig. 1.2 Stage structure when we have one stage per day. For simplicity we use only two outcomes of the random variable per day

Assume we model electricity demand and production by the day over a week. The model has seven time periods. We could create a model with equally many stages, which would require that we for example modeled the following sequence (where we assume that decisions for a given day are made *before* demand for that day is revealed). See also Fig. 1.2:

- Decisions made Monday morning for Monday production
- Demand information for Monday arrives
- Decisions made Tuesday morning for Tuesday production
- Demand information for Tuesday arrives
- . . .
- Demand information for Saturday arrives
- Decisions made Sunday morning for Sunday production
- Demand information for Sunday arrives

Modelled this way, the model has as many time periods as it has stages. But we might find that this is a bit detailed. We may realize that the model will be rerun every day, so in fact, only the first set of decisions will be implemented (in the example the Monday decisions). So we instead create the following model, also illustrated in Fig. 1.3:

- Decisions made Monday morning for Monday's production
- Demand information for Monday arrives
- Decisions made Tuesday morning for Tuesday's and Wednesday's production
- Demand information for Tuesday and Wednesday arrives
- Decisions made Thursday morning for production the rest of the week
- Demand information for Thursday to Sunday arrives

This latter model still has seven time periods, but only three stages, Monday, Tuesday, and Thursday mornings (it is a question of taste if you also count the final "Summing up" as a stage). So we see how the arrival of information has been aggregated, and the number of stages reduced. Finding a good trade-off between

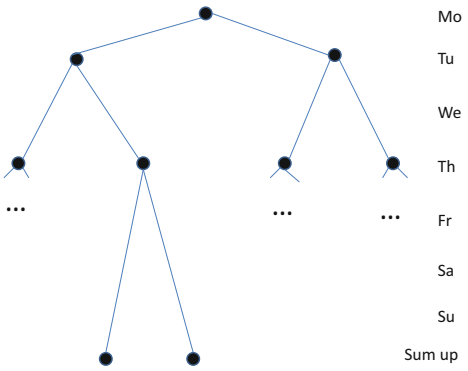


Fig. 1.3 Stage structure when we have fewer stages than time periods. For simplicity we use only two outcomes of the random variable per stage

time periods and stages is often crucial when modeling as it has consequences for model quality, data collection, and solvability of the model.

Inherently multistage models can become inherently two-stage models if we add a stage with a qualitatively different decision framework. For example, we can add an initial investment (buying vehicles, building warehouses or factories) and end up with an inherently two-stage model with the first stage as an investment stage and the remaining periods form a second stage (or multiple stages) that provide information about operational costs and benefits of the initial investment. Chapter 6 discusses a technique for modeling the *end-of-horizon* issue in multi-period decision problems; we shall also see there that many operational stages become embedded into a single, horizon, stage. It is important to note that the inherent structure is a modeling statement, not related to the actual number of stages (decision points) in the model.

Inherently multistage models are, in our opinion, less suited for stochastic programming than their inherently two-stage counterparts. This is particularly true if the models are set in a kind of steady state environment. Stochastic programs focus on the transient aspect of decision-making. A financial investor, for example, may find herself stuck with bad investments, high transaction costs (possibly caused by low liquidity in her assets), or violations of financial regulations. In this case a stochastic program might be well suited. But an investor who finds herself well-placed may be better served by a steady state model.

1.8 Distributions—Do They Exist and Can We Find Them?

It is tempting to reply "no" and then "yes" on these questions. That obviously requires some explanation. In our view we must be careful when we approach this issue. On the one hand, there is the distinction between the existence of a distribution and our ability to find out what it is. On the other hand there is the distinction between repeated and one-time events. And there are also cases where the situation

we face will become a repeated event, but so far we have no experience with the case. And we should never forget that we are *modeling*, that is, we are attempting to simplify the real world, so as to learn about it. That obviously also implies modeling, and hence simplifying, the description of the random phenomena.

Whatever view we have on how random phenomena should be approached, in stochastic programming we are always talking about the future. Apart from such as odds at a casino or the probability of winning in Lotto, it is an *assumption* that whatever we might know about the past will be relevant for the future. So even if we have a futuristic view on life, and even if we have loads of data, how do we know that the future has anything do to with that past? Although the authors of this book believe we can say something useful about the future, and that mathematical modeling is a useful tool, we should never forget that this is an assumption or a belief. We can certainly test, in most cases, if the past was a good description of the future in the past, but that is as far as we can get. The assumption that this is still true cannot be tested.

You may say that we have the whole machinery of statistics to help us say something about the future. But using statistics this way exactly implies assuming that the past does describe the future. There have been a fair number of financial crises over the last few years showing that at critical points in time, the past does *not* describe the future well.

> We can never test if our description of future events is useful or good. This remains an assumption or a belief.

The view we take here is in some sense more pessimistic than the view of many that object to stochastic programming due to the need for distributions. However, the issue is not how optimistic or pessimistic we are, but how we move from this point.

As we have seen in this chapter, sensitivity analysis, what-if analysis, and the like may not deliver what they promise. We miss out on important aspects of robustness and flexibility in the solutions. If you face a problem with an important random variable, even if you know very little about the actual distribution, we would strongly advice to set up a stochastic program with two or three scenarios (three may be wise so that you get mass in the middle). That way your solution will change dramatically in structure (provided your feeling about the importance of the parameter was right), and you will learn a lot about your problem. This might mean solving a problem about twice the size of the original problem, something you can probably handle. Not to do this because you do not know the distribution is in our view a very bad choice. The chance is that by using a distribution you will obtain a reasonable feeling about the medicine, if not the dosage.

It is a peculiar argument that uses the lack of precise knowledge of the distributions as an argument for using deterministic modeling. Putting all the probability mass in the mean of the distribution is certainly neither right nor good.

Alternatively, many prefer to use worst-case analysis to analyze the problem. They may work with intervals and come up with results that do not depend on the actual distributions. This might sound tempting, but in our view is often not good modeling. Worst-case analysis over an interval will normally amount to putting all the mass in the end points of the interval, which we find to be a very peculiar distribution. The results might be very sensitive to the choice of interval, and that is not good if the interval is subjectively chosen.

Supports of random variables in stochastic programming are also most of the time subjective. But you are aware of this issue, of course! The use of low probabilities in the tails offsets this to some extent. Soft rather than hard constraints (that is, we put the constraints into the objective function with a penalty for violation) makes tail effects much less serious.

Distributions, combined with soft constraints, should be used to learn about the decision problem.

Information we might have about a phenomenon, even if that information is subjective or circumstantial, should be used. If we have good data, we should use them. If we have expert opinions, we should use them. If all we have is our own humble understanding of the problem, we should still create distributions so as to obtain a feeling for what constitutes good solutions to our problem—what are the missing options. We should be able to find the right medicine, even if the dosage might be off. After all, we are modeling, and we search for knowledge and understanding.

1.8.1 Generating Scenarios

The generation of scenarios from distributions (wherever they might come from) requires some care. This might seem to be in contradiction to our very subjective, loose, view on distributions expressed above. While we view the creation of a distribution as a modeling issue, we still need to be sure that what we solve is what we think we solve. Otherwise our understanding from the modeling may be driven by errors in the scenario generation.

This is why we, in a book on modeling, present two chapters on scenario generation. Even if you are not at all interested in the subject as such, you should be concerned about the relationship between your model (particularly the description of uncertainty) and the stochastic program you solved. If your interest is the

model and real decision-making, and not merely the ability to solve stochastic programs, you should be concerned that the solutions you are studying (and possibly implementing!) are not driven by how you make scenarios, but by the actual problem formulation.

1.8.2 Dependent Random Variables

If you replace all random variables by their means, you end up with a deterministic optimization problem. Much of this book is about why that might not be a good idea, especially when there are opportunities to learn about the information and take actions. Another, related sense in which expected values are misleading is that it obscures *dependencies* among these random variables. There is nothing in a deterministic model that even tries to represent dependencies.

In addition, you will often see in the literature on stochastic modeling, especially in steady state modeling, that random variables are assumed independent. Stochastic programming has the tools to be able to come to grips with dependencies in an explicit way.

1.8.2.1 Risk Management

Dependencies arise in risk management in a big way. The canonical example is as follows:

- *Risk Management.* You are responsible for managing risk in a large company, with several divisions and lines of business. They sell different products and services whose profitability depends on many different factors. How do you go about managing the overall risk, rather than the individual risks?

Let us look at a very simple example. Assume we have a company consisting of two divisions, each facing the same distribution of profit in the next period:

Profit	−1	0	2	4	5
Probability	0.1	0.2	0.4	0.2	0.1

The expected value is 2, which is considered OK. But there is an understanding that the negative profit is problematic, so each division decides to buy insurance against it. If the profit is −1, which happens with 10% probability, they will receive 1 from the insurance company. The expected return on the policy is 0.1, for which they will have to pay 50% extra, that is 0.15. The profit distribution (including the insurance premia and payouts) becomes:

Profit	−0.15	1.85	3.85	4.85
Probability	0.3	0.4	0.2	0.1

The mean is now 1.95, a reduction by 0.05 but giving us an acceptable risk profile.

At the company level, the mean has been reduced from 4 to 3.9 obtaining the following profit distribution:

Profit	−0.3	3.7	7.7	9.7
Probability	0.3	0.4	0.2	0.1

So, in particular, it does not matter if each division insures its risk or the mother company insures both divisions at the same time.

Is this right? Think about it before you proceed.

Did you realize that what we assumed above is that the two profit streams are perfectly correlated? In particular that the bad cases would occur together? Is that reasonable? It might be of course, but such perfect correlation is rare. What if we instead assume that they are uncorrelated? It takes a bit of work, but you should easily enough be able to calculate it as follows:

Profit	−2	−1	0	1	2	3	4	5	6	7	8	9	10
Percent	1	4	4	8	16	4	26	4	16	8	4	2	1

But this is less risky! The company can lose 2 with a probability of 0.01 and 1 with a probability of 0.04. So a policy bringing this up to zero would have an expected payoff of 0.06. If there is still a premium of 50%, the cost will be merely 0.09 rather than 0.30, but more importantly, the loss in expectation will have been reduced from 0.1 to 0.03.

But what if the profits are perfectly negatively correlated? In this case there is no risk to the mother company at all! So if the divisions made their own risk adjustments, there will be a sure loss of 0.1, with no benefit at all.

> Correlations and more complicated forms of co-variation are crucial to capture when planning. Otherwise all measures of risk can be totally off.

What do we learn apart from the fact that correlations are important? We learn that it is very dangerous to look at an activity part by part. What looks risky may not be risky at all! Almost all projects and activities are part of a larger portfolio. Risk must be measured at portfolio level. Even projects with negative expectations may be profitable for a company! (Why?)

It is our view that much literature treating risk fails to capture the portfolio effect. The net result is a poor understanding of risk. Stochastic programming is well set

up to model any kind of co-variation. This is contrary to many other tools (but not contrary to all tools of course). We shall discuss this to some extent in Chap. 3.

1.9 Characterizing Some Examples

Let us look at four typical decision problems under uncertainty that we have seen so far to analyze their main characteristics.

Truck Routing on Page 15—To Be Discussed in More Detail in Chap. 7

Stage structure:	This problem is inherently two-stage with a finite (but large) number of stages as the plan will last for six months.
Random variables:	Demand of the different commodities is assumed to be random.
Objective function:	The goal here is to minimize expected cost. It is reasonable to assume that the average behavior of the problem will be observed over a reasonable period of time, and that we therefore are not risk averse.
Why not deterministic?	A deterministic model would produce routes that are not very flexible with respect to variation in demand. We shall see that in Chap. 7.

Sports Event on Page 15

Stage structure:	This problem is inherently two-stage with, principally, an infinite number of stages as you have no plans to stop this activity.
Random variables:	The weather, particularly the probability of rain.
Objective function:	Most likely you will minimize expected costs as you will observe the average performance of your business. But if each event is very large and the events are far apart in time (even if regular) you would likely want to model some risk aversion.
Why not deterministic?	In a deterministic model it would either not rain (in which case the optimal solution is to do nothing) or rain (in which case you would choose the cheapest way to get the event under roof or simply give up.) Other solutions which trade-off the two situations (and possibly take into account weather forecasts) would not be valued.

Electricity Production on Page 18

Stage structure:	This problem is inherently multistage with, pricipally, an infinite number of stages as you have no plans to stop this activity.

Random variables:	Both supply and demand are uncertain, most of this is driven by the weather.
Objective function:	You would likely minimize expected costs here as these are repeated operations where you will observe the average behavior.
Why not deterministic?	Deterministic models will either produce solutions that cannot handle noise when it occurs or which ends up overly pessimistic (and hence costly). A major issue is that many production units have longer startup/closedown times than the time interval over which wind can be predicted.

Risk Management on Page 23—To Be Discussed in More Detail in Chap. 3

Stage structure:	This problem is inherently multistage with, pricipally, an infinite number of stages as you have no plans to stop this activity. If you were considering some major changes in procedures or some major investment that would change the whole portfolio, this problem might be inherently two-stage.
Random variables:	Market developments and success of individual projects.
Objective function:	Here you wish to model risk aversion probably caused by some of the projects having correlated profits. If all divisions and project had uncorrelated profits, there would be no risk management problem to attend to.
Why not deterministic?	Risk management makes little sense in a deterministic world.

1.10 Alternative Approaches

The rest of this chapter will be used to outline other approaches than stochastic programming to decision-making under uncertainty. Our goal is to put the different ideas into perspective and see where they agree and where they differ. We would also like to point out strengths and weaknesses of each. We shall only present basic ideas and point to principal difficulties, modeling wise, from using the ideas.

1.10.1 Real Options Theory

This theory comes from economics, rather than operations research, which is the home of the other methods we cover in this book. The starting point in options theory is "what is it worth," rather than "what shall we do." Despite this difference in focus, stochastic programming and options theory have a lot in common. Value

is often a by-product of a stochastic program, and options problems incorporate decision-making (albeit in a simplified way).

The term *real* options theory stands in contrast to *financial* (not imaginary!) options theory and indicates that we are talking about real (physical) operational decisions, rather than the implicit decisions embedded in a financial security. Like financial options, a large part of the value in operational decisions lies in the possibility to make better decisions in the future.

All option valuations can be formulated as stochastic programs (not necessarily in an efficient way). But there are many problems that lend themselves to stochastic programming where option theory cannot be used.

Building an Oil Platform

For a concrete example, assume you are consulting on the construction of an oil platform. One of the issues is: What size should the platform deck have? The bigger the deck, the larger the platform legs, and the more expensive the platform. By adding extra deck space, you buy an option, however: You make it possible later on to add more equipment. Should the field produce more sand than expected, you can add an extra unit for sand removal. Should the platform produce more oil than expected, you can add a production unit, and maybe you end up producing so much water that you need an extra unit for that. So, on the one hand you have a certain cost, on the other hand an uncertain gain. You buy the option if the expected gain is higher than the certain cost. The stochastic programming approach to pricing such an option takes a different route than is followed in finance courses (although our conclusions will easily be predicted by students of finance, at least for the stripped-down formulations that resemble options pricing problems).

In stochastic programming, our emphasis will be on the practical side—using the stochastic programming approach makes it much easier to incorporate details of actual operations. On the other hand, in stochastic programming we cannot use the more powerful computational approaches and simplifications made possible by options pricing's use of continuous time stochastic processes. In real options settings, however, operational details may actually be much more important to solution quality than the representation of the probability space. In financial applications the use of continuous time stochastic processes gives quickly obtainable solutions, and that is often highly valued. It must be noted that in many real optional settings (and even some financial ones) the price of ignoring operational details has a great impact on the business value of the solutions.

Oil-Well Abandonment

The canonical problem in real options theory is oil-well abandonment. An oil well consumes electricity (among other services) and produces a profit that depends on the price of oil. As the oil field ages, the quantity of oil declines at a rate that is fairly well-understood (usually there is geological information or at the very least, information from nearby wells). Abandoning the oil well costs money and is irreversible for all practical purposes, at least for this problem setting. The question is *when to abandon the well.*

The deterministic approach to this problem is to apply net present value (NPV) analysis. According to NPV analysis, the well should be abandoned when the expected discounted future profits equal the cost of well abandonment. This is correct if there is no uncertainty. However, the price of oil (and of electricity) is variable. NPV neglects the profit that could be obtained if the price of oil shoots up faster than the cost of operations. This potential profit is called "the option value" of the well; for further information on the real options approach, see Sick [81].

For us the concept of option is included in the notion of *recourse actions*, which embody the potential to respond when information has been observed. The additional and very important detail of *when* to execute a recourse action is a topic that lies beyond the scope of this book.

1.10.1.1 Finding Recourse Options

In Sect. 1.6 we discussed robustness and flexibility, and we pointed out how flexibility and robustness related to one another. We showed, for example, how buying a (real) option to face a future uncertainty is a robust action leading to future operational flexibility.

But the options we defined were all very explicit and well defined. These can come about in two different ways: problem knowledge and by solving a stochastic program. Once an option is defined, it can (if feasible) be evaluated using options theory or be entered into a stochastic program as a possible decision. Let us first look at two cases where problem knowledge has been used to define options.

Scheduling and Real Options
Scheduling is an important and economically significant planning problem all on its own. Airlines have large divisions for handling real-time disturbances or shocks in their operations. They cancel flights, reroute passengers, switch crews or planes, deadhead crew, put passengers in hotels, or whatever is necessary to minimize the cost of a disturbance. The cost includes both monetary outlays and bad publicity which impacts future revenues. The costs are substantial, and keeping costs low is a major goal. Substantial effort is invested into developing effective methods to recover from shocks.

Recovery Costs Versus Planned Operating Costs
There is an obvious, but still much overlooked, problem in the above approach. As the ability to find good (deterministic) solutions increases, the planned costs (the costs incurred when everything goes as planned) become lower and lower. However, implicit in the formulations used to find schedules, we almost always find the assumption that the world is deterministic. Of course, nobody actually thinks this is the case. But that is beyond the point: The models are deterministic, hence leading to solutions with typical knife-edge properties as discussed on page 6.

Such solutions normally do not have good expected behavior relative to surprises, such as delays and breakdowns. We saw that in our newsmix example on page 4, and it is also outlined in Higle and Wallace [40] and Wallace [85]. Hence, the

expected recovery costs might become very high. So the combination of a world-class scheduling department and a world-class recovery department may lead to a mediocre total performance, because they fit so badly together.

Real Options in Airline Scheduling
In the airline industry we now see an increased understanding for these issues. Ehrgott and Ryan [19, 20] have described a model where the robustness variable is extra ground time, and the optimization amounts to finding an optimal distribution of ground times. This way they try to maximize the probability that minor delays have no effects at all. Rosenberger et al. [76] have gone one step further and analyzed the potential effects of limiting the number of airplane types at certain medium and small airports.

Adding constraints will increase the planned costs. But this makes the schedules, as seen by the customers, much better as they more effectively withstand shocks. The quality comes from the fact that with fewer airplane types, the ability to switch crews and planes on short notice increases, thereby increasing the day-by-day operational flexibility. In addition this approach reduces maintenance costs. Robustness and flexibility in the airline industry are also discussed in Ball et al. [4].

Telecommunications
From telecommunications we can follow the ideas of robust network designs back to Suurballe and Tarjan [82] who discussed two-connected networks, that is, networks where each pair of nodes is connected by at least two paths not sharing arcs. That makes the network well protected against arcs failing since information can be routed along different paths. Even better performance can be obtained by going to 3-connected and generally n-connected networks.

What characterizes these ideas from airline scheduling and telecommunications is that they are obtained by problem knowledge. The model defines what we are to check. This is also in line with what options theory offers (although the above articles do not relate to options theory), namely an answer to the question: "Here is an option, what it is worth?" With this ability to find values, we can also compare options. However, options theory cannot be used to *find* options.

Option theory cannot be used to find options, just to evaluate them.

This lack of ability to *find* optional actions is particularly serious. Options theory can never tell the absolute quality of a solution. One may find the best of four options (correctly), but cannot know if there are substantially better options available. We refer to Chap. 7 for the analysis of a case where defining the *recourse options* is the critical step.

1.10.2 Chance Constrained Models

Chance constrained models are models that replace a constraint of the type $ax \leq b$ by one that requires the constraint to be satisfied with a certain probability. For example, that demand must be met 95% of the time or that there must be less than 1% chance of going bankrupt. Often these statements really represent what we want. The 1% chance of bankruptcy might be a company policy that we are forced to follow. The same may be true of the 95% chance of being able to deliver—a 95% service level.

We may wish to ask: What is the optimal service level relative to the costs or benefits? Although most businesses find it convenient to measure goals in terms of service level goals, it is important to keep in mind that there must be a well thought through trade-off between the value of a high service level and the costs of implementation. By using chance constraints we are pushing the process of checking if the present level is indeed optimal into the background.

The problem with a chance constraint is that the optimization will attempt to ignore bad cases unless measures are taken to address them. A service cost minimization model, say, will try to achieve the 95% service level as cheaply as possible. It will therefore try to handle the 95% easiest cases and leave out the 5% most costly ones. The optimization may disregard some investments that could reduce the costs of the remaining 5%. This problem cannot be overcome by parametrically varying the service level requirement. It can only be addressed by bringing all cases into the objective function where the costs and benefits can be compared.

> Solutions to chance constrained models may have very bad overall performance unless measures are taken to model costs and benefits arising from the cases that are excluded.

There are two major types of chance constraints: Individual and joint. In the first case we put requirements on an individual constraint, like the probability of going bankrupt. In the second case, we make statements about several constraints at the same time. An example could be that there must be a certain probability that the demand for *all* our products is met. We do not discuss algorithms in this book, but we trust you understand that to set up such a model you will need an ability to handle dependence in random variables, as discussed in Sect. 1.8.2. The solution needed to obtain a given service level will normally strongly depend on how the individual demands (random variables) depend on each other.

1.10.3 Robust Optimization

Robust optimization is an alternative framework for modeling decision problems where some parameters are uncertain. When you look at the literature you will see that there are many flavors of robust optimization, so perhaps there is a bit of confusion about what robust optimization means. In this section we give brief descriptions of the flavors that we know about.

The main use of the term "robust optimization" in recent years has been applied to methodologies in mathematical programming to analyze the dependence of solutions on parameters. A volume by Ben-Tal et al [1] explores this topic in great mathematical detail. Many problems are highly sensitive to particular parameters—including problems that are widely recognized as canonical examples in various fields. General techniques to identify problems with sensitivities and systematically explore the impact of errors in estimating sensitive parameters are a very good *starting point* to address the sorts of issues that we are concerned about here in this book.

The term "robust" has been applied to certain model formulations in stochastic programming, in which the "second stage" collects error terms from soft constraints and includes them in the objective, possibly computed using a nonlinear function.

A third flavor, due to Bertsimas and Sim [8], models a simple two-point error distribution for each parameter independently and asks the question: what is the relationship between the objective function value and the probability of finding a feasible solution for this objective value? Let us call this last flavor "stochastic robust optimization" to distinguish it from the other ones.

1.10.3.1 Stochastic Robust Optimization

Here is a rough overview of stochastic robust optimization as presented in the original paper. Many of these ideas have later been developed. Our point is *not* to make statements about all these models in general but to use this paper to give you an example of how to think about modeling whenever you face a new approach.

The idea is to look at the uncertain variables as enemies that try to make life hard for you. But it is, according to the authors, not very likely that all the enemies act at the same time. So the authors define a parameter Γ_i, where the indexing is over rows, which controls how many of the uncertain variables in a given row act against you at the same time. In this way, the optimal solution $\hat{x}$ becomes a function of Γ, such that if you implement $\hat{x}(\Gamma)$ you are guaranteed that if up to Γ_i uncertain variables in row i work against you at the same time, even if they all take on their most problematic values, your solution is still feasible. This way you can, in principle, trace a frontier between the protection level Γ and the cost corresponding to $\hat{x}(\Gamma)$.

To do this you obviously have to define an interval over which each uncertain variable can vary. The authors use the nominal value plus and minus a constant. Although this is very similar to a support for a random variable, and certainly may be exactly that, in general it is "simply" an interval over which you wish to be

protected. The choice of interval will normally be subjective. The authors are not using random variables as such and, in particular, do not use distributions. This is certainly an advantage in the sense that we do not have to specify the distributions. On the other hand, as we shall see shortly, this advantage definitely comes at a cost. Also, in our view, if you have partial information about the distribution it is bad modeling practice not to use it. In our view, even worst-case analysis should be over that part of the problem you do not understand. But here the modeling philosophies differ. You have to take a stand on this question.

Stochastic robust optimization is both different from and similar to stochastic programming. It has advantages and disadvantages. One advantage that it shares with stochastic programming is that the problem type does not change radically from its deterministic original. Stochastic linear programs are linear programs, just larger. Stochastic robust optimization versions of linear programs are also linear programs but stochastic robust optimization problems do not grow in size as fast as stochastic programs in the number of stochastic parameters, which is a huge advantage over stochastic programs. On the other hand, the original formulation of stochastic robust optimization problems allows only simple stochastic distributions and does not really address the complex features involved in the choice of distributions such as utility, risk and reward trade-offs, dependence between random variables, and so forth.

Robustness Is Useful

Let us look at stochastic robust optimization formulations in the light of robustness and flexibility as defined by us in Sect. 1.6. Their goal is to create a first-stage decision (these models have only one stage) which is robust in the sense of being able to withstand shocks. However, there are three related problems here.

How to Achieve Robustness?

The first potential problem is that setting up a model with only robust variables (only one stage) may be very costly. Very often robustness at one level is connected to flexibility at a lower level to achieve good overall performance. A goods transportation company facing serious randomness in demand may still allow itself a robust (in our terminology) publicized schedule because it has operational flexibility in the routing of goods. Remember that such a schedule will be of the type "There is an overnight service from New York to Chicago," but it will not say *how* the goods are sent (for example if the goods are reloaded along the route). A schedule that tries to handle all the uncertainty by using robustness, without taking into account rerouting of goods (typical second-stage variables) will probably be very expensive and based on having far too many trucks available. Planning routes without taking into account the actual operational flexibility simply is too conservative. A reason for the computational simplicity of stochastic robust optimization is exactly the lack of stages.

Since the models do not have stages, all the decisions are "robust" in our terminology. It is important to be aware of this different use of the term "robust."

Feasibility
The second problem with many robust formulations is that they lead to solutions that are feasible only with a certain probability, just as for chance constraints. In fact, chance constraints and stochastic robust optimization results in a very similar formulation and interpretation. (Although we do not pursue this thought, it seems that one could view stochastic robust optimization as a nice way of implementing a certain restricted family of chance constrained problems.) Consequently, many of the observations we made concerning chance constrained problems apply here, too.

Interval Sensitivity
The third problem is the choice of interval for the uncertain parameters. The solutions you obtain may be very sensitive toward these choices, and in most cases they are rather subjective. Normally, also supports of random variables in stochastic programs are subjectively chosen, but the extreme effects are offset by the willingness to use probabilities, expressing that these extreme values are not very likely. Also, when low probabilities are combined with stages, less extreme effects are achieved in stochastic programs than in these robust formulations.

Combining these three issues, we see that we have to be careful with stochastic robust formulations. Not all forms of robustness are economical. The robustness achieved may be very costly in its own right, and the expected performance may be very bad. In repeated operations, it is usually the expected performance we care about.

As with chance-constraints formulations, stochastic robust formulations may produce solutions with a bad expected performance.

Remember that the purpose of this section was to illustrate how you might want to think about a new approach you see to find out if the underlying assumptions really fit your needs. Our discussion concerns only one approach (though a rather popular one), and the comments do not generally apply to robust formulations. Whenever you see (or develop) a new approach you need to be careful and critical about what it really means. Maybe it fits you perfectly, but maybe it does not. That is your responsibility to find out.

1.10.4 Distributionally Robust Optimization

A merge between stochastic programming and stochastic robust optimization has emerged, and it has been named *distributionally robust optimization*. In our view, this is sound modeling. The starting point is a stochastic program with stages and distributions. That overcomes many of the arguments we made against stochastic robust optimization. But in the spirit of stochastic robust optimization, a worst-case

approach is taken with respect to whatever is not known about the distributions. Or to put it differently, we start by taken all the distributional information we have (actual values, nonnegativity, bounds, moments, correlations . . .) but then pick the worst possible distribution, according to our objective function, having these properties. So we use all information available, but not more.

In general, this procedure is probably not numerically, or even modeling wise, feasible. But there are surprisingly many special and interesting cases that are easily solved. So if you are somewhat mathematically inclined, there are many interesting papers and theses to be written down this lane.

1.10.5 Stochastic Dynamic Programming

We have already pointed out that stochastic programming primarily is about transient decision-making. Stochastic dynamic programming, on the other hand, is normally focused on long-term steady state behavior. We define *state variables*, such as inventory levels, present staffing, the weather—whatever is relevant for our problem—and then for each possible *state*, that is for each possible value of our multi-dimensional state space, we calculate what should be done. So we end up with *decision rules* or *policies* saying that if the state is so-and-so, we should do so-and-so. This in contrast to stochastic programs which give us specific suggestions for decisions in specific situations.

> Stochastic dynamic programming formulations focus on long-term steady state, not transient behavior.

If you have a problem that fits the dynamic programming framework, then it is a very efficient approach. But dynamic programming is rather limited with respect to what stochastics it can handle, and it cannot handle very complicated constraints and problem definitions.

1.10.6 Rolling Horizon

Let us end this introduction with something that is not really an approach to handling an uncertain future. But as it is sometimes refereed to as being a way to handle uncertainty we want to mention it and give it its right place in our toolbox. Assume you have a multi-period problem, such a production inventory problem, i.e., a problem that runs over many periods, and where in each period there is production (that you need to determine) and demand, and where products produced but not sold are carried forward as inventory. There are all sorts of variants of this problem, but that is not our focus just now.

For simplicity, assume there is a known and finite number of periods before the problem ends. Let us set up one deterministic model where all demands are replaced by their means, and one truly inherently multistage model. Assume that we have generated scenarios in some way for the stochastic model. We have already argued against the deterministic model, so that is not the point of this section. Rather, assume that we ask: We know the weaknesses of the deterministic model, but if we still use it (maybe for numerical reasons), how much will we lose? What is the expected loss of using this deterministic model? Let us assume the model is a cost minimization model. Let f_d be the optimal objective function value for the deterministic model, and f_s the corresponding value for the stochastic model. Is $f_s - f_d$ a good measure of how much we lose? Think about it.

The answer is no. The reason is as follows: Any practical person responsible for such a production problem, even if he used this deterministic model to determine production, would rerun the model each period, and when doing so, taking into account what was actually sold in the previous period, and not just using the expected demand (which was in the model). So even though the model is deterministic when it looks into the future, it takes (realizations) of the stochastic demands into account when it looks backward; it updates the state of the world. This is called rolling horizon. For each period (stage), the deterministic model is rerun based on observations. So f_d is not an estimate of the costs using the deterministic model. But what is a good estimate?

Assume for a moment that the scenarios used in the stochastic model are the true distribution of demand. This is just to make the discussion a bit easier to follow. Realizing that using a deterministic model does not make the problem deterministic, we understand that we must test how the model would function in the stochastic world. One way would be to use the scenarios from the stochastic model, but each time a decision must be made, take the one from the deterministic model. Would that result in a good cost estimate for using the deterministic model?

The answer is again no. And the reason is that the user of the deterministic model would use it in a rolling horizon framework. So to get the right estimate of the costs using the deterministic model, we must set up the scenario tree, and whenever a decision needs to be made, we must run the deterministic model using the inventory at that point in time. Disregarding the fact that deterministic models are (usually) used in a rolling horizon framework will make the deterministic model look even worse than it really is.

So rolling horizon means to always look back at what has happened but does not, in its own right, imply looking properly forward to what might happen later. So it is indeed correct to use a rolling horizon framework, but it does not overcome the weaknesses of deterministic models we have outlines in this chapter. Note that even stochastic models, if used repeatedly, should be set in a rolling horizon framework when evaluated or compared.

A rolling horizon frameworks should always be used when evaluating models, but it is not a way to overcome the problems of deterministic modeling.

Chapter 2
Information Structures and Feasibility

That man is prudent who neither hopes nor fears anything from the uncertain events of the future.
– *Anatole France*

As was illustrated in our example in Chap. 1, it is not straightforward to pass from a deterministic to a stochastic formulation. We need to rethink the whole model, very often by changing both variables and constraints. Although many reformulations may make sense mathematically, they may in fact be rather peculiar in terms of interpretations. The purpose of this section is to discuss some of these issues, partly in terms of an example. The goal is not to declare some formulations generally superior to others but rather to help you think carefully about how you rewrite your problems in light of uncertainty.

2.1 The Knapsack Problem

As an example, let us look at the knapsack problem. The problem is simple to write down, namely

$$\begin{aligned} &\text{maximize} && \sum_{i=1}^{n} c_i x_i \\ &\text{such that} && \begin{cases} \sum_{i=1}^{n} w_i x_i \leq b \\ x_i \in \{0,1\} \ , \ i = 1, \ldots, n \end{cases} \end{aligned} \tag{2.1}$$

A. J. King, S. W. Wallace, *Modeling with Stochastic Programming*, Springer Series in Operations Research and Financial Engineering,
https://doi.org/10.1007/978-3-031-54550-4_2

where

c_i is the value of item i,
w_i its weight, and
b is the capacity of the knapsack.

The goal is to fill the knapsack with as many valuable items as possible, but without breaking the weight limit. Of course, w_i might also be viewed as "size," in which case the volume of the knapsack is the capacity in question.

Assume now that the weights are uncertain, so that, in fact, we are facing a vector of random variables $[\boldsymbol{w_1}, \ldots, \boldsymbol{w_n}]$. How are we to interpret this situation? The first question to be asked is always:

> What is the inherent stage structure, and how many stages are there?

A clear clue to the stage structure is *when will we learn the weight of an item*? Obvious suggestions are:

1. We learn the weight *of each item just before* we decide whether or not to put it into the knapsack.
2. We learn the weight *of each item just after* putting it in.
3. We learn the weight *of the full set of items just after* we decide what items to put in.

The first two interpretations can lead to both *inherently two-stage problems* and *inherently multistage problems* with as many stages as there are items. The last interpretation will normally lead to an inherently two-stage formulation, in that we first decide on which items to put in and only thereafter observe if they in fact fit. An additional aspect of stage structure is how potential infeasibilities are handled. After all, even though weights are uncertain, the capacity of the knapsack is fixed.

Let us list some potential ways of handling these stage structure questions. For the moment we limit our discussion to the inherently two-stage cases where all items are picked (or listed in a specific order) before learning anything about their weights.

2.1.1 Feasibility in the Inherently Two-Stage Knapsack Problem

1. We may require that the chosen set of items always fits into the knapsack.
2. We may list the items in a certain order and pick them up until we come to one that does not fit. Then we stop. (So the decision is the list.)
3. We may do as above, but if a later item fits (as it is light enough) we take it.

4. We may list the items and keep adding items until we have added an item that did not fit. We then pay a penalty for the "overweight."
5. We may pick a set of items, such that if the items do not fit into the knapsack after we have learned their weights, we pay a penalty for the total overweight.
6. We may pick a set of items of maximal value so that the probability that the items will not fit is below a certain level.

There are certainly more variations, but let us stop here for now. You should think about what these cases imply before continuing reading. One way to structure the analysis is to ask a central questions:

When Do We Learn the Weights of the Items?
In the Case 2 above, we list the items and stop putting them in when we find one that does not fit. This implies that we learn the weight of an item *before* it is actually put into the knapsack. Case 3 is a variant of the Case 2, since it amounts to stopping when one does not fit and then continuing down the list until we find ones that do fit. Case 4 implies that we need to actually put the item into the knapsack before observing its weight. So the second and the fourth case represent quite different interpretations of when we learn the weights. In Case 5 we learn the weights after we have decided on the selection of items.

Note that putting items into the knapsack is a very passive action in these cases, since we have already decided in which order items are going to be picked up (for Case 5 we simply pick them all). If we wish the order to depend on the actual observed sizes, we end up with an inherently multistage formulation, which we shall discuss a bit later.

Cases 2, 3, and 4 result in inherently two-stage models, since we define the lists before we start putting items into the knapsack. Stage one is to find the list of items, while stage two is to passively put them into the knapsack until the stopping rules are satisfied. But to set up the lists you have to anticipate the different situations that can occur. Hence, the models will be multistage with an inherent two-stage structure.

Case 5 is also an inherently two-stage formulation, leading to an inherently two-stage model. The model will have only two actual stages as there is no question of ordering the items.

Case 6 results in a chance constrained formulation that we shall discuss shortly.

Case 1, namely to require that the chosen set of items, always fits into the knapsack corresponds to a worst-case analysis. Since we need to find a set of items that always fits in, we can replace the random variables $\boldsymbol{w_i}$ by their maximal values. Of course, this formulation makes sense only if there is an upper bound on the weights.

The worst-case analysis corresponds to a very "pessimistic" view on the world, namely we can *never* accept overweight. Whether or not this is reasonable is a *modeling* question. We must look at the situation in which the model is used and ask ourselves if it is really the case that we cannot handle an overweight item.

If we plan to put in an item, is there nothing we can do to get it to "fit"? If the knapsack is a truck, and the items are the loads we plan to send, could we not send

some items with the next truck? Maybe we could put a package in the passenger seat? Maybe we could send it by mail? Requiring feasibility in this way is extremely strong, and we must be sure we really wish to imply a worst-case situation.

Finally, of course, our estimates of the maximal weights may be incorrect. This may lead to an actual situation with overweight, even if the model said it would not happen! Then what will we do? Will the world end? Will the company go broke for sure? Probably not. But if we *can* handle overweight when it really happens, why did we model the problem as if it could not be allowed to happen under any circumstance? You should really be able to answer these questions if you wish to use worst-case analysis.

2.1.2 The Two-Stage Models

So, some of the models, while being inherently two-stage, are multistage in nature. Those that are not are the worst-case analysis (which is always particularly risky to use if the worst-case is not well defined) and the last two cases; the one with a penalty for total overweight and the one looking at the probability of overweight. Let us first look at a penalty case. Let $\mathcal{S}$ be the set of scenarios describing the uncertainty in an appropriate way, then we get

$$\begin{aligned} \max \quad & \sum_{i=1}^{n} c_i x_i - d \sum_{s\in\mathcal{S}} p^s z^s \\ \text{such that} \quad & \begin{cases} \sum_{i=1}^{n} w_i^s x_i - z^s \le b & \forall s \in \mathcal{S} \\ z^s \ge 0 & \forall s \in \mathcal{S} \\ x_i \in \{0,1\} & 1 = 1, \ldots, n \end{cases} \end{aligned} \tag{2.2}$$

where d is the unit penalty for overweight. A more general penalty could be a function $f(z^s)$ describing a non-linear dependence on the total overweight. This model might be good or bad, depending on how well it describes our case. It has, however, a clear interpretation, as it has a clear information structure (we learn about the weights after having decided which items to use), it has a clear description of the goal (to maximize the value of the items selected minus the expected penalty for overweight) and it says what happens if we get it wrong—we pay a penalty. The penalty may mean exactly that, a financial penalty for being wrong. But it may also mean such as a cost for sending an item with a later truck, the extra cost of using a competitor, or possibly a rejection cost.

This formulation can be viewed as replacing a constraint with a penalty, since it can be written as

$$\max_{x_i\in\{0,1\}} \quad \sum_{i=1}^{n} c_i x_i - d \sum_{s\in\mathcal{S}} p^s \left[\sum_{i=1}^{n} w_i^s x_i - b \right]_+ \tag{2.3}$$

where $[x]_+$ is equal to x if $x \geq 0$, zero otherwise. We call this a *penalty formulation*.

Case 1 in our listing, the worst-case analysis, can be formulated as follows:

$$\begin{aligned} \max_x \quad & \sum_{i=1}^{n} c_i x_i \\ \text{such that} \quad & \begin{cases} \sum_{i=1}^{n} w_i^{\max} x_i \leq b \\ x_i \in \{0,1\} \qquad 1 = 1, \ldots, n \end{cases} \end{aligned}$$

There is not much to say about this one. It is very pessimistic and the model is, technically speaking, deterministic. Of course, in some cases, this is exactly what we need, so the model may be appropriate. Note, however, as mentioned above, that this is a very sensitive model unless $w_{\max}$ is well understood. Therefore, although mathematically well defined, this model may be pessimistic *and* risky at the same time. So, in general this model is hard to defend. On the one hand we claim that the items *must* fit in the knapsack, on the other hand, we risk that they do not unless we know $w_{\max}$ precisely. Note that the average behavior of the solution coming from this model might be bad, as we do not attempt to control it.

2.1.3 Chance Constrained Models

Let us pass to Case 6 in our listing, a model that tries to get around the problem of feasibility by requiring that the items fit in the knapsack with a certain probability. The standard model in this case within stochastic programming is a chance constrained model. It would take the following form:

$$\begin{aligned} \max_{x_i \in \{0,1\}} \quad & \sum_{i=1}^{n} c_i x_i \\ \text{such that} \quad & \begin{cases} \sum_{s \in W(x)} p^s \geq \alpha \\ W(x) = \{s : \sum_{i=1}^{n} w_i^s x_i \leq b\} \end{cases} \end{aligned} \tag{2.4}$$

where α is the required probability of feasibility. This model is clear with respect to the objective function, and also how to treat infeasibilities.

Chance constrained models say nothing about what happens if we have overweight. In a sense, a chance constrained problem is a slight relaxation of a worst-case analysis. In the truck example, this model means that we plan which items to put on the truck and we require that our plans work out α percent of the time. What we do when the items do not fit is not clear. Perhaps we ship them off on another slower channel, and the probability level is in fact a service level for the quick transfer.

If the monetary cost is reasonably well connected to α, we also have controlled the costs. But a danger with this formulation is that the costs associated with lack of feasibility may be connected to the size of the violation, not just the probability, and a chance constrained model does not have any control over this aspect of the solution.

2.1.4 Stochastic Robust Formulations

Chance constrained problems (particularly with discrete variables) can be hard to solve. That is especially so for problems more complicated than what we discuss here. Stochastic robust optimization (discussed in Sect. 1.10.3.1) offers an alternative. The worst-case analysis discussed above is a type of robust formulation—we are looking for the most profitable solution that is always feasible. However, there are more sophisticated formulations with a trade-off between loss in income and increase in probability of feasibility. Also in these models we totally disregard what lack of feasibility actually costs. Let us write the model in a way that is reasonably easy to understand, although this is not the format we would use to solve it. Let there be N items available, such that item i has a weight coming from a symmetric distribution over $[w_i - \hat{w}_i, w_i + \hat{w}_i]$, and let Γ be an integer between 1 and N. Let $\mathcal{N} = \{1, 2, \ldots, N\}$. The following formulation will give us the set of items with maximal value under the constraint that if at most Γ of the random weights work against us, we still have a feasible solution, i.e. we still can fit the items in the knapsack, whatever values these weights take. The general probability of feasibility is also high. Bounds are complicated but are given in the underlying paper by Bertsimas and Sim [8].

$$\max_{x\in\{0,1\}} \quad \sum_{i=1}^{n} c_i x_i$$

$$\text{such that} \quad \begin{cases} \sum_{i=1}^{n} w_i x_i + \psi(x,\Gamma) \le b \\ \psi(x,\Gamma) = \max_{S\subset\mathcal{N},|S|=\Gamma} \sum_{i\in S} \hat{w}_i x_i \end{cases}$$

As we increase Γ we get closer and closer to a worst-case situation. Also in this model, there is no statement about what actually happens when items do not fit. As before, that might or might not be a problem.

A major reason these robust models do not control average profits is of course that they do not use probabilities at all, just supports of the random variables (or generally: intervals over which we wish to be protected). In its own right that may be good, but it certainly carries with it potential surprises when solutions are implemented. The combination of considering neither the costs of overweight nor the probabilities thereof is not without potential risks.

Note that the worst-case formulation given earlier, and the case above with $\Gamma = N$ are not the same, as the x-variables are not defined exactly the same way.

2.1.5 The Two Different Multi-Stage Formulations

When making the knapsack problem stochastic, there are two different multistage settings, one is inherently two-stage, the other inherently multistage. In one case we ask for a decision rule of the type: items $i_1, i_2, \ldots, i_k$ have been added and the weights turned out to be $w_1, w_2, \ldots, w_k$, so which item should I put in next? This problem is inherently multistage. This is an extremely difficult problem if you require optimality.

The alternative multistage question is the following: Give me the (ordered) list of items and follow certain rules for stopping. Here, we do not change our minds on the order as we fit them in based on observations, but we take uncertainty into account when setting up the list. This formulation is inherently two-stage. A good test for you is to formulate this latter problem as a decision-tree problem under the assumption that each item has only a limited number of possible weights. Try it!

2.2 Overhaul Project Example

This example is taken from Anderson, Sweeney, and Williams, Section 10.4 [2]. Let us first repeat the problem and the analysis as given in the reference. We have an overhaul project with five activities, labeled A through E. The example is as given in Table 2.1. The activity-on-arc network for this little project is given in Fig. 2.1.

Table 2.1 Activity names, expected durations, and immediate predecessors

Activity	Description	Immediate predecessor	Expected Duration (Days)
A	Overhaul machine I	–	7
B	Adjust machine I	A	3
C	Overhaul machine II	–	6
D	Adjust machine II	C	3
E	Test system	B,D	2

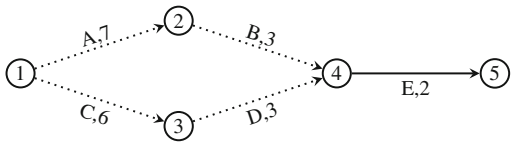


Fig. 2.1 Example network for overhaul example. Each arc is labeled with its activity code and duration

Table 2.2 Maximum possible reduction and corresponding costs

Activity	Description	Maximal reduction	Cost per day
A	Overhaul machine I	3	100
B	Adjust machine I	1	150
C	Overhaul machine II	2	200
D	Adjust machine II	2	175
E	Test system	1	250

The longest path through this network is given by the sequence of activities A, B, and E with a completion time of 12. This path is called the *critical path* as any delay on an activity on this path will delay the whole project. The partial sequence C-D has a slack of 1, indicating that if either activity (but not both!) is delayed by 1 day, the project completion will not be delayed.

Suppose that it has become evident that the overhaul project must be completed within 10 days. With the data presented in Table 2.1, this is not possible, and the company is willing to invest money to reduce the project duration. The company may reduce the durations of the activities for a cost. For each activity, the reduction costs and the maximum possible reductions are given in the Table 2.2.

Let y_A denote the number of days by which we reduce the duration of activity A, which will cost $100y_A$. With variables for the other activities similarly defined, the following linear program can be used to determine the minimum cost of reducing the project duration to 10 days, as required. The variable x_i denotes the time at which event i begins. For example, event 4 is the time when both activities D and B have finished, and hence, E is ready to start.

$$\begin{aligned} \min \quad & 100y_A + 150y_B + 200y_C + 175y_D + 250y_E \\ \text{such that} \quad & \begin{cases} x_2 \geq x_1 + 7 - y_A \\ x_3 \geq x_1 + 6 - y_C \\ x_4 \geq x_2 + 3 - y_B \\ x_4 \geq x_3 + 3 - y_D \\ x_5 \geq x_4 + 2 - y_E \\ x_1 = 0 \\ x_5 \leq 10 \\ y_A \leq 3 \\ y_B \leq 1 \\ y_C \leq 2 \\ y_D \leq 2 \\ y_E \leq 1 \\ x, y \geq 0 \end{cases} \end{aligned} \tag{2.5}$$

The minimal investment necessary to complete the project within 10 days is 350 and is obtained by setting $y_A = 1$ and $y_E = 1$ with all other investments zero. Note that all activities are now on a critical path. This is a typical result from this type of

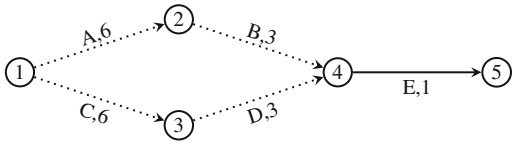


Fig. 2.2 Example network after investments in activity durations. All paths are critical

Table 2.3 Probability distribution for activity durations relative to the mean

Activity	Description	Probability of deviation from expected value		
		−1	0	+1
A	Overhaul machine I	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$
B	Adjust machine I	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$
C	Overhaul machine II	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$
D	Adjust machine II	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$
E	Test system	0	1	0

model. Our investment results in a large number of critical paths, and the extreme case is what we observed here: All activities sit on critical paths. The new network is given in Fig. 2.2.

2.2.1 Analysis

How are we to interpret this result? At a cost of 350 we have reduced the project duration to 10, as required, and all activities have become critical. What does that mean? Does it mean that any delay in any activity will cause a project delay? The answer depends on the nature of the data presented in Tables 2.1 and 2.2.

Suppose that the activity durations given in Figs. 2.1 (before investments) and 2.2 (after investments) are not really deterministic. Although many possibilities exist, let us assume that the given numbers are expected durations, and that the distributions are independent and as indicated in Table 2.3. That is, suppose that with the exception of activity E, all activity durations will be their expected values, plus one of three values: {-1, 0, 1}, with all values being equally likely. Activity E is assumed to have a deterministic duration.

Given that activity durations are random variables, the project duration is also a random variable. Since this is such an easy network, we can calculate the distribution of the project duration exactly. We begin by calculating the distribution of the duration of activities A and B together and then do the same for C and D. Event 4 will then take place when the last of (A and B) and (C and D) is finished. Finally, we add the duration of E.

Since the duration of A (after investments) is 5, 6, or 7, while that of B is 2, 3, or 4, activities A *and* B will have finished after 7, 8, 9, 10, or 11 days. The distribution can be found by examining all possible combinations of the completion times:

Duration of A	5	5	5	6	6	6	7	7	7
Duration of B	2	3	4	2	3	4	2	3	4
Duration of A *and* B	7	8	9	8	9	10	9	10	11

Each of these events occurs with probability $\frac{1}{3}\frac{1}{3} = \frac{1}{9}$. Hence, the distribution for the duration of A *and* B is given by

Duration of A *and* B	7	8	9	10	11
Probability	$\frac{1}{9}$	$\frac{2}{9}$	$\frac{3}{9}$	$\frac{2}{9}$	$\frac{1}{9}$

This is also the duration of C *and* D. Event 4, which corresponds to the start of activity E occurs when all of the first four activities have finished. Formally, we may state that

$$\text{Start time for event 4} = \max\{\text{Duration of A and B, Duration of C and D}\}$$

To calculate the distribution of this maximization, we simply look at all 25 combinations of durations of (A and B) and (C and D). The project duration is simply the start time for event 4, plus the duration of activity E (which is 1 day). Thus, we get the following distribution for the duration of the project.

Duration of project	8	9	10	11	12
Probability	$\frac{1}{81}$	$\frac{8}{81}$	$\frac{27}{81}$	$\frac{28}{81}$	$\frac{17}{81}$

The expected project duration is therefore 10.642, well above the required duration of 10. In fact there is a probability of 56% that the project will take longer than 10. Hence, it seems that our investment of 350 has not brought the duration down to 10. It has not even brought the expected duration down to 10.

At this point you should worry about the sequencing of decisions, and the information that is available as individual decisions are made. Our initial analysis assumed that all activities ended up with their expected duration and that all decisions were made initially. That is, our initial analysis focused exclusively on one scenario that has probability $\frac{1}{81}$ of occurring.

2.2.2 A Two-Stage Version

In the above discussion we solved a problem where all activities were assumed to have average duration in order to find a possible investment. Thereafter, we checked how this solution/investment would behave in the random environment. But, when finding the solution, we did not consider the uncertainty in the activity durations. As a result, one of the effects was that we did not even achieve an expected project duration of ten.

Let us now reconsider the investment, this time recognizing the uncertainty in the activity durations. Suppose that the investment must be determined before the actual durations can be known. This is inherently two-stage. Let d_A^s be the duration of activity A in scenario s, and let the other durations be defined accordingly. Since the duration of activity E is not subject to uncertainty, we do not define such an entity for this activity. Let the investment variables, y, be defined as before, and let x_i^s be the time of event i if scenario s occurs. We have now $3^4 = 81$ equally likely scenarios corresponding to the 81 possible realizations of durations of the activities A through D.

A question that occurs at this time is what we are to mean by the project taking 10 days. Is it going to be an average of 10 days, or is 10 days a hard constraint? Or, maybe, 10 days is the goal, but, at a penalty, we are allowed to be late? In reality, of course, this type of constraint is not hard. We can never guarantee that a project cannot be late. We could certainly find an investment, that with our 81 scenarios, guaranteed that we were never late, but reality will always be different. Hence, let us instead assume that if we are late, a penalty of 275 per day is incurred.

Note first that if we had added the possibility of being late at a penalty of 275 to the deterministic problem, the solution would not have changed, as it is cheaper to invest in reducing activity durations than to pay the penalties. Also, note that the expected cost associated with the deterministic solution is now the initial investment of 350 plus an expected penalty for being late of 210, at a total of 560. So the initial cost estimate of 350 was far off the actual cost.

The following two-stage model will minimize the expected cost of achieving a project duration of 10, *provided all investment decisions are made before the project starts, and we are allowed to be late*. Lateness in scenario s is measured by t^s.

$$\min \quad 100y_A + 150y_B + 200y_C + 175y_D + 250y_E + \frac{275}{81}\sum_{s=1}^{81} t^s$$

$$\text{such that} \quad \begin{cases} x_2^s \geq x_1^s + d_A^s - y_A & \text{for all } s \\ x_3^s \geq x_1^s + d_C^s - y_C & \text{for all } s \\ x_4^s \geq x_2^s + d_B^s - y_B & \text{for all } s \\ x_4^s \geq x_3^s + d_D^s - y_D & \text{for all } s \\ x_5^s \geq x_4^s + 2 - y_E & \text{for all } s \\ x_1^s = 0 & \text{for all } s \\ x_5^s - t^s \leq 10 & \text{for all } s \\ y_A \leq 3 & \\ y_B \leq 1 & \\ y_C \leq 2 & \\ y_D \leq 2 & \\ y_E \leq 1 & \\ x, y \geq 0 & \\ t^s \geq 0 & \text{for all } s \end{cases} \tag{2.6}$$

What distinguishes problem (2.6) from problem (2.5) is found in the representation of the constraint on the project duration. In (2.5), this was represented by

$$x_5 \leq 10.$$

In (2.6) we have

$$x_5^s - t^s \leq 10,$$

indicating that the duration should be at most ten but can be allowed to be longer. Furthermore, all constraints that are used to calculate the event times, such as

$$x_2^s \geq x_1^s + 7 - y_A$$

are now indexed by scenario. Solving this we find $y_A = 1$ as the only investment. The cost is 100. The expected penalty for delays is as high as 455, yielding a total of 555, a reduction of five from the expected value of the deterministic solution. (As this is just an artificial example, the numbers themselves are not important, the main point is that we get a different and cheaper solution.)

What happened here is that as we realized explicitly that the world is stochastic and that delays are in fact feasible (at a cost), we ended up investing much less initially to reduce the project duration. Instead, we preferred to pay the penalty for being late. With $y_A = 1$ as the only investment, there is a probability that event 4 takes place later than time 8 (so that we finish the whole project later than 10) of 89%. Hence, the penalty is incurred in 89% of the cases. As 89% of 275 equals less than 250 (the unit investment cost in activity E), we prefer the penalty cost. So in this case, there was flexibility in *waiting*. Instead of securing the project duration initially, it is better to wait and see what happens.

This formulation is an inherently two-stage formulation, leading to a two-stage model, as we first make investments, then observe the activity durations, and finally (stage 2) calculate the project duration and delay costs.

2.2.3 A Different Inherently Two-Stage Formulation

The ideal formulation of this project scheduling problem is to take into account the actual float of information. Initially, we must decide on y_A and y_C. Then, y_B is decided when activity A is finished, and y_D when activity C is finished. However, we do not know which of these events will occur first. Modeling wise, this creates a lot of difficulty if we are to formulate the problem as a stochastic programming problem. It makes us unable to define stages. Stage 1 is to define y_A and y_C, but what is stage 2? It is to define y_B if A finishes before C, but to define y_D if B finishes before A. And which of these will happen first depends on both the randomness and our first-stage decisions. Stage 4 is in any case to determine y_E.

Hence, let us analyze a somewhat simpler case. Let us assume that investments in activities A through D must be determined initially, but that activity E can wait until the activity is to start. This will make y_E scenario dependent. In addition, we can be late at a penalty.

$$\min \quad 100y_A + 150y_B + 200y_C + 175y_D + \frac{250}{81}\sum_{s=1}^{81} y_E^s + \frac{275}{81}\sum_{s=1}^{81} t^s$$

$$\text{such that} \quad \begin{cases} x_2^s \geq x_1^s + d_A^s - y_A & \text{for all } s \\ x_3^s \geq x_1^s + d_C^s - y_C & \text{for all } s \\ x_4^s \geq x_2^s + d_B^s - y_B & \text{for all } s \\ x_4^s \geq x_3^s + d_D^s - y_D & \text{for all } s \\ x_5^s \geq x_4^s + 2 - y_E^s & \text{for all } s \\ x_1^s = 0 & \text{for all } s \\ x_5^s - t^s \leq 10 & \text{for all } s \\ y_A \leq 3 & \\ y_B \leq 1 & \\ y_C \leq 2 & \\ y_D \leq 2 & \\ y_E^s \leq 1 & \text{for all } s \\ x, y \geq 0 & \\ t^s \geq 0 & \text{for all } s \end{cases} \tag{2.7}$$

It is a challenge to guess what the solution will be based on what we have learned so far. Without calculations, you should be able to see that the investment will be $y_A = 1$ as the only investment. Furthermore, since investments in E can be delayed until we are ready to start the activity, we will choose to invest in E if we start later than time 8. This is so since the cost of investing in E is lower than the penalty cost of being late. If we are ready to start activity E later than time 9, we shall invest in a one unit decrease in E and take the rest of the delay as a penalty. The total expected cost is down to 533.

2.2.4 Worst-Case Analysis

An obvious way to make sure the duration is at most ten (in fact the only way) is to perform a worst-case analysis. We then resolve (2.5), but with maximal durations rather than average durations. Hence, the durations of activities A, B, C, and D will increase by one. The result will be $y_A = 3$, $y_B = 0$, $y_C = 0$, $y_D = 2$, and $y_E = 1$ with a total cost of 900. Of course, in this case the expected delay cost is zero. But be aware that this is assured only if our description of uncertainty is correct. Hence,

worst-case analysis can be both conservative and risky at the same time; we are careful, pay a lot to be *sure* that we are always feasible, but then, due to errors in estimating the data, we are not so sure after all. Worst-case solutions do not handle measurement errors.

2.2.5 A Comparison

There are a few points to be made here. The first concerns feasibility. In the deterministic model, it was reasonable and meaningful to say that we had to finish in ten time periods. But if we kept that requirement in the stochastic setting, we were brought to a worst-case analysis. If we have an inherently two-stage model (i.e., all investments are made before the project is started), the only way to guarantee a duration of ten time periods is to plan as if everything is against you. We saw that the cost would be 900. Very often, such strict interpretations of feasibility are not reasonable. Instead it is necessary to ask if a constraint is *really* hard? Very often the answer is no. If the softness that is therefore brought into the model can be described by penalties, we end up with a *recourse* model. We gave two examples of such models. In a two-stage setting with penalties, we ended up with expected costs down from 900 to 555. If we, in addition, allowed the second stage to also contain a genuine investment, the expected cost dropped to 533. The latter drop is simply caused by the new investment opportunity being cheaper than the delay cost. If we solved the deterministic model, it claimed that the investment cost would be 350, whereas, in fact, the total expected cost was 560. With a strict interpretation of feasibility, the deterministic solution was infeasible with probability 0.56.

In addition to the issue of feasibility, we observe that there is again a value of delaying decisions. We saw that the total expected cost depended on how we defined these possible delays. We cannot say that one model is better than the other unless we actually know what is the real decision context. But we see how the modeling choices affect decisions and costs.

2.2.6 Dependent Random Variables

In Sect. 2.2.1 we assumed that all the random durations were independent. We then found that using the investment from the deterministic model $y_A = y_E = 1$, the expected cost was not 350 as indicated by the deterministic model, but rather 560 if the late penalty was set at 275 per day. The probability of being late was as high as 56%. If the durations had instead been correlated, the deterministic model would have been the same, hence the investments the same, but the expected costs would be different, and so would the probability of delays.

Let us see what would happen if activities A and B were perfectly negatively correlated and activities C and D perfectly positively correlated. These are of course extreme assumptions but they serve to illustrate a few points.

First, the duration of A and B would be deterministically equal to 9. The perfect negative correlation (correlation coefficient of -1) has removed the uncertainty on that path. As always, a negative correlation has helped us control the variation. A negative correlation of -1 is of course rather special (as it implies we have only one random variable, not two), but any negative correlation between the durations of A and B would reduce overall uncertainty and be useful to us.

> Negative correlations are always potentially useful, and you should think carefully about how they might help you.

For the other path, a perfect positive correlation would cause the duration of C and D to be 7, 9 or 11, each with a probability of $\frac{1}{3}$. This is worse than in the deterministic case in the sense that while the probability of being above 9 (causing the project to have a duration above 10) has stayed at $\frac{1}{3}$, the expected delay, given that there is a delay, has increased. Here we see that positive correlations have caused us trouble, as they normally do.

So the starting time for Event 4 is again the maximum of the duration of these two paths plus one, leading to the following distribution for the project duration.

Duration of project	10	12
Probability	$\frac{2}{3}$	$\frac{1}{3}$

So the probability of being late is now 33%, and the expected project duration 10.67. The expected cost is now 533, down from 560.

The point of this is to understand that there is not just the question of stochastic/deterministic, but also *which* stochastic model we are facing. Correlations (and other measures of co-variation) have no counterparts in the deterministic modeling, making several rather different stochastic settings being represented by the same deterministic model.

So the question is not that one type of stochastics (like uncorrelated durations which we started out with) is "better" than others. Rather, it is that the effects of solving a deterministic model depend on the stochastic setting. We saw that when we changed from uncorrelated random variables to correlated ones (in this one specific way), expected costs went down, expected duration up, and the probability of delay did not change. Again, this is not good or bad, it is simply an observation telling us that issues that do not even come up in deterministic modeling, such as co-variation, can be important in valuing the actual performance of a deterministic model.

2.2.7 *Using Sensitivity Analysis Correctly*

In this example we assumed that there was a penalty cost of 275 per day if we were late. We simply used it as a number. Is that appropriate in the light of the discussions in this book? You may want to think about that questions for a few minutes!

If the number is a specific estimate of the cost of delay based on market activities, like a contractual penalty, *and nothing else*, such as lost image, we may face two situations:

- If this is an estimate of what the future value of the penalty will be, based on its present value, then it should be treated as a random variable, and our approach is not valid in light of our own discussions: We have used expected values instead of the actual distribution.
- If this a known entity, which we know will not change during the life of the project, then our approach is indeed valid.

But, most likely, even if there is a contractual penalty for lateness, there is also a question of lost goodwill, lost reputation. And the size of that loss is not really known. It is anybody's guess. So 275 is our guess, our chosen value for the overall costs. But it is a guess, a choice, not because we are facing a random variable, but because it is up to us to define the penalty: the penalty is in its own right a policy parameter for the company. If that is the case, our approach is appropriate but should possibly be accompanies by a parametric analysis on the level of the penalty.

So what is actually the right answer depends on the setting. However, most likely, this is a case where parametric analysis is appropriate because the penalty is like a decision variable: It is up to us to set it. We leave it to you to check what would happen if the penalty is slightly different from 275 (in particular a bit higher).

2.3 Summing up about Feasibility

The three examples, together with the newsvendor example from Sect. 1.2, show many different aspects of modeling stochastic decision problems. However, much of the discussion, one way or another, concerns feasibility. Let us try to sum up what we have seen.

What we notice here is that although many deterministic models can be made stochastic, the steps cannot be automated. We have to think about what the constraints actually imply in the stochastic setting, and we must be particularly concerned about the handling of feasibility. Very often feasibility is not as strict as we imply by our modeling, and using penalties is better.

Feasibility in this context has two components. First, constraints that seem reasonable in a deterministic setting turn into worst-case analysis in a stochastic environment. Although that might be what we want, it is rarely the case. The difficulty is simply that in a deterministic setting, where parameters normally are

expected values, the constraints seem reasonable, even if they in fact represent goals that might be violated. After all, the model only handles the average case.

But in a stochastic setting, if the constraints actually represent wishes or goals, and violations can be allowed, though possibly at a high cost, the constraints should be moved into the objective function with a penalty. Except for what we might call book-keeping constraints—inventory out equals inventory in plus production minus sales—you should have very strong reasons to use constraints in stochastic models.

It is customary to call constraints that remain, technical speaking, constraints, for *hard* constraints, while those moved into the objective function are called *soft* constraints. Our claim is that from a modeling perspective, most constraints are soft.

The second component of feasibility is that many, if not all, deterministic models lack a proper stage structure. Even though the deterministic model may have time periods included, the variables are not properly adjusted. A side effect of that is that when the information structure of an event tree is added to a problem, the variables must be redefined. For example, in the newsvendor example of Sect. 1.2 we had the same variable for orders and sales. This might be fine in a deterministic world where you never produce something you will not need, but in a stochastic model you must realize that production and sales belong to different stages of the model and cannot be represented by the same variables. A more straightforward example was seen in the overhaul project example in Sect. 2.2 where we had to put a scenario index on some variables. That amounts to defining new variables, even though we continued to use the old variable names.

Chapter 3
Modeling the Objective Function

If you do not know where you are going, every road will get you nowhere.
– *Henry Kissinger*

A goal without a plan is just a wish.
– *Larry Elder*

The objective function of a mathematical program is what an optimization procedure uses to select better solutions over poorer solutions. For example, if the objective is to maximize profit, then the procedure tries to move in the direction of solutions that increase profit while still remaining feasible. But when the profit depends on a parameter that is uncertain (like prices tomorrow), then the notion of maximizing profit is no longer very simple.

Distribution of Outcomes

The broadest perspective you could take on this question is that your decision, once taken today, results in a distribution of outcomes. Your choice amounts to a choice of one distribution from a whole family of outcomes distributions "parametrized" by your decision. If you ignore the randomness in the problem (as many do, but of course you are not one of these!), your procedures will still select one distribution from this family—but you will be unable to control that choice. You need some way to inform the optimization process to select distributions with favorable characteristics.

> Making a decision can be viewed as choosing one particular outcomes distribution over all others.

A. J. King, S. W. Wallace, *Modeling with Stochastic Programming*, Springer Series in Operations Research and Financial Engineering,
https://doi.org/10.1007/978-3-031-54550-4_3

But what is a favorable characteristic of an outcomes distribution? This is a question with no simple answer. There have been many scientific avenues of inquiry into this issue. There is still no one way to look at the question and there remain many strange paradoxes. We will be satisfied to indicate some practical concepts that can be used to select better outcomes distributions over poor outcomes distributions.

3.1 The Knapsack Problem, Continued

Let us first review the knapsack problem in its soft-constraint formulation

$$\max_{x_i \in \{0,1\}} \quad \sum_{i=1}^{n} c_i x_i - d \sum_{s \in \mathcal{S}} p^s \left[\sum_{i=1}^{n} w_i^s x_i - b \right]_+ \tag{3.1}$$

Consider a solution $\hat{x}$. Unless we have a large capacity, there will be a collection of sample points $s \in \mathcal{W}(\hat{x})$, representing scenarios where the combined weight of the selected items turns out to be larger than the maximum weight allowed. So the distribution of objective function values will be a random variable:

$$V_s(\hat{x}) = \begin{cases} \sum_i \left(c_i - d w_i^s\right) \hat{x}_i + db & \text{if } s \in \mathcal{W}(\hat{x}) \\ \sum_i c_i \hat{x}_i & \text{otherwise} \end{cases}$$

To analyze a solution to a stochastic program, you will need to examine the distribution of the objective value and ask yourself: What are the features that we are concerned about?

Perhaps the most important practical features of the distribution are the expected value and the upper and lower quantiles—for example, the 10% and 90% quantiles. Are these what you expected? Should the penalty d or the weight limit b be adjusted?

The main point we wish to make here is that it is a big challenge to design an objective function that fully captures all the desired features of the outcomes, and it is very rare to get it right the first time!

Soft constraints partition the underlying sample space into favorable and unfavorable outcomes. It is perhaps worth examining this partition to learn whether this is what was intended.

Consider a knapsack problem with two different customers. If the total weight of accepted items from the two customers were very different, this might be a bad outcome. For example, let us say that items in the set O_1 come from the first customer class and items in the set O_2 come from the second customer. We can track this difference by calculating the expected difference of the weights of the excluded items

$$\sum_{s\in\mathcal{W}(\hat{x})} p^s \left| \sum_{i\in O_1} w_i^s \hat{x}_i - \sum_{i\in O_2} w_i^s \hat{x}_i \right|$$

When this difference is too large, it might be bad for customer relations! Now, think for a minute—how could you change the problem to address this issue?

Consider also the expected contribution of each item to the objective function, which can be modeled as the profit for including the item less than its expected contribution to the overweight situation.

$$\left(c_i - d \sum_{s\in\mathcal{W}(\hat{x})} p^s w_i^s \right) \hat{x}_i$$

Slicing and dicing the contributions by item attributes may lead to important insights into other features that need to be controlled. The main point is that you should look carefully at the outcomes distribution and verify whether its properties were intended.

3.2 Using Expected Values

The most commonly used way of comparing two outcomes distributions is to compare the expected values. By "using expected values," we do not mean that you are using a single point (the mean) as the realization of the random parameters. Rather we mean that you are optimizing the expected value of the outcomes distribution.

Sometimes we choose an expected value criterion because this is the simplest and most convenient choice. But in many cases it is also the right thing to do. Here we shall outline the most common arguments for maximizing expected profit or minimizing expected cost.

You Observe the Expected Value

Here you face a situation that will be repeated over and over again, and then simple repetition favors the expected value criterion.

For example, the newsvendor of Sect. 1.2 makes a fixed order that will be used daily, say for the next year. In such a case, even if there are severe variation in daily costs, it is still reasonable to minimize *expected* cost. The law of large numbers takes over. Your annual result may be so close to the expected cost that you would not care about the difference.

If you are planning for outcomes that will be repeated many times, then it is likely correct to use expected value as the criteria for choosing an outcomes distribution.

Before rejecting the mean value as an objective criterion, you should dig into the operational details of how the decisions will be used in operations. For example, even in the case where the newsvendor revises the order on a weekly basis, the average of the weekly costs over a year will also approximate the mean value—even though the variation in the weekly costs may be quite large. (How would you go about verifying this statement?)

The Company has Share-Holders

Making decisions under uncertainty in a corporate setting has some special features that have been the subject of a deep and extensive literature. We will base our discussion here on a paper of Froot and Stein [26]. The basic idea is that a public company has share-holders, who themselves are making decisions under uncertainty about how many shares they wish to hold in which company. Very often share-holders want to be exposed to risk, since risk also means opportunity. The question is what sort of risk management should be pursued by the company?

Let us suppose that the company faces a major internal decision with risky outcomes. For simplicity of argument, let us also assume that even in the worst of circumstances the company will not go broke. (The issue of bankruptcy is discussed below.)

Here are a couple of questions:

- Would you be risk averse in this situation, or go for the expected value?
- Would you be willing to buy insurance to avoid the worst outcomes?

The way to answer this argument, according to [26], is to put yourself in the shoes of one of your share-holders. Let us represent the share-holder by a wise lady who understands that most of the risk in your profit is caused by your technology choices. She understands that companies may have different solutions to the problem at hand, and your success (or failure) depends on which solution your customers end up preferring.

To hedge this uncertainty concerning technologies, she buys equally many shares in your competitor's company. From her perspective, there is no longer any risk—except those that relate to her exposure to the market for your and your competitor's products.

What will happen now if both companies recognize the risks they are facing? For example, suppose they both decided to insure their technology risks? Well, our wise lady is still facing no market risk, but now, whichever technology wins, she gets less! She will not be very happy with this decision.

If only one of the companies reduces risks, it is even worse. Possibly without knowing it, she is now facing risks, since the symmetry between the two companies is gone *and* the expected values of her investments have gone down. So to assure investors like her that she can safely invest in your company, she must be assured that you do not behave in such a way that increases the risks for her.

A company with share-holders should maximize expected profit and leave risk adjustments to the owners, except for those risks that can be hedged more cheaply by the company itself.

Some risk reducing measures are only available to companies, as a result of taxation rules, perhaps, or as a result of access to markets. If so, companies can and should be risk averse in those respects. It is crucial that investors realize that this is being done. Many companies make statements to this respect in their public reporting. If you run into this problem in a practical setting, it is wise to consult with financially trained people. Our point is that you become aware that even in decisions where you do not observe repeated outcomes it may not be appropriate to reduce all the risk you face.

Our wise lady's strategy is related to a very popular statistical arbitrage called "pairs trading," in which investors take a short position in one competitor and an offsetting long position in another. The investor is not exposed to the success or failure of the underlying market but will make money on a temporary deviation that supposedly should revert to the statistical long-run mean. This kind of gamble is really a by-product of how large corporations are managed. There really is no way to reward management for outcomes based on whether a given technology is good or bad. The actual practice is to reward "steady earnings growth" of the sort that can only be achieved by holding a diversified portfolio generating the products and services sold by their sales forces. If two large companies hold diversified portfolios and comparable brands, then taking bets on statistical properties of their earnings is a plausible arbitrage strategy.

The Project is Small Relative to the Total Wealth of a Company or Person

Even if the variance of the income from an investment is very high relative to the mean, this variability usually does not create concern if the numbers happen to be small. When someone tells you: "The cost will be two or three dollars, I am not sure which," you probably say: "OK, I don't mind." You are not concerned despite the fact that one estimate is 50% higher than the other.

On the other hand, if the same person tells you: "Oh, it will cost two or three million dollars," you probably would hesitate, even if the ratio between the two cost estimates is the same. Why? Probably because in the latter case, the amounts are substantial relative to your total wealth.

Use expected value if the relative importance of the project is small.

This is actually a variant of the first argument about observing the expected value. If a project is small relative to a company's total wealth, it probably has a very large number of these projects, and it will observe the expected value (but now the expectation is over projects and not over outcomes within a single project).

If none of these arguments apply, it is probably time for you to think properly about what risk actually means in your case, and if you should be worried about it.

3.3 Penalties, Targets, Shortfall, Options, and Recourse

When soft constraints are incorporated into an objective function, the part of the objective function that models the soft constraint will have a certain shape. Again, look at the knapsack problem (3.1). The soft-constraint part of the objective is the expected value of a piecewise linear function of the excess weight. It is zero below the capacity b and has slope d above the capacity. This piecewise linear function has many names and it appears in many contexts.

Penalty Functions
In the language of linear programming, such functions are sometimes called *penalty functions*. The penalty function has two parts. One is the *target interval*, which in the knapsack case is the interval below the capacity, namely $(-\infty, b]$. The second part is the *penalty rate*, namely the rate at which the penalty accumulates as the target is missed. This rate may be an actual cost of responding to the penalty, but more usually it is a modeling approach that is used by the modeler to shape the outcomes distribution. An example for the knapsack problem can be found in Fig. 3.1.

Targets and Shortfall
One can use penalty functions to indicate a desire to reach a target. Although there is a similarity to a penalty formulation of a soft constraint, a target formulation is

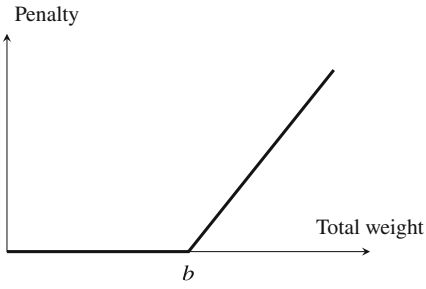


Fig. 3.1 Penalty function for the knapsack problem

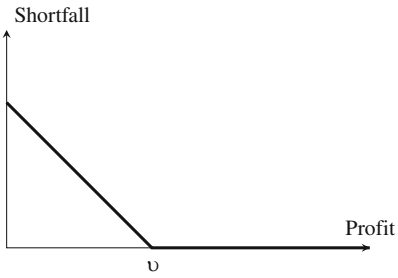


Fig. 3.2 Example of shortfall function with a target v

not really a soft constraint since the act of selecting a target reshapes the outcomes distribution. If you prefer outcomes x^s above a given target v to outcomes below the target, then the simplest criterion that measures this preference is called the *shortfall measure*

$$\sum_{s\in\mathcal{S}} p^s(v - x^s)^+. \tag{3.2}$$

Consider the function

$$h(y) = \begin{cases} 0 & \text{if } y < 0 \\ y & \text{if } y \geq 0 \end{cases} \tag{3.3}$$

This is a piecewise linear function with slope 0 below zero and slope 1 above zero. An illustration of a penalty function can be found in Fig. 3.2. It is not hard to see that

$$\sum_{s\in\mathcal{S}} p^s h(v - x^s) = \sum_{s\in\mathcal{S}} p^s(v - x^s)^+. \tag{3.4}$$

This function compares two outcomes distributions by looking at the expected values over the region below the target v. So in the situation where x^s represents the return of an investment portfolio over some time horizon, then perhaps we would like to avoid outcomes with higher values of $h(v - x^s)$. If we are in a maximizing frame of mind (most investors are), then we could maximize the following expression:

$$\max_{x\in X} \left\{ \sum_{s\in\mathcal{S}} p^s x^s - \sum_{s\in\mathcal{S}} p^s h(v - x^s) \right\} \tag{3.5}$$

where X is some set that constrains our portfolio choices. What are we maximizing here? We are maximizing a piecewise linear function, which, computationally, is not too hard. What would it do? Well, if we had two outcomes distributions with

similar means, this method of choosing outcomes would prefer the one with a larger *conditional* expectation over the outcomes that lie above the target v.

These types of shortfall measures are very useful in applications involving uncertainty. Target shortfall measures are a natural way to describe differences in outcomes distributions, and the optimization technology required to solve them is readily available. We shall find many examples of these in this book.

Options

Penalty functions model a cost that is incurred when certain underlying events occur. In finance, a contract with terms that incur a cost or produce a payment depending on the occurrence of a future event is called an "option." A typical type of option is a call option, which grants the owner the right to buy an underlying security at a fixed price, called the strike price, at some fixed dates in the future.

The owner of the call option has a choice on an exercise date. If the price of the underlying is above the strike price, then the owner can buy the underlying security for the strike price and then sell it at the market price. The owner's profit equals the difference between the market price and the strike price. On the other hand, if the market price is below the strike, then the owner does not have to do anything. The option payout is a function that is zero below the strike and increases linearly with slope one above the strike as in Fig. 3.3.

Can you see that an option payout (Fig. 3.3) looks like a penalty function (Fig. 3.1) with target equal to the interval above the strike price and rate equal to one? Also note the similarity with the shortfall function (Fig. 3.2). Many penalty formulations can be framed in terms of call and put options because penalties are essentially invoked when a stochastic value goes above or below a target.

Recourse

Another term that applies to penalty formulations is *recourse*. A recourse model describes actions that lead to a future cost or benefit in response to future events. In the case of a penalty formulation, the recourse model just calculates the penalty (such models are called "simple recourse" in the literature). But a recourse action could be more complex. A recourse model can minimize the impact of a bad event

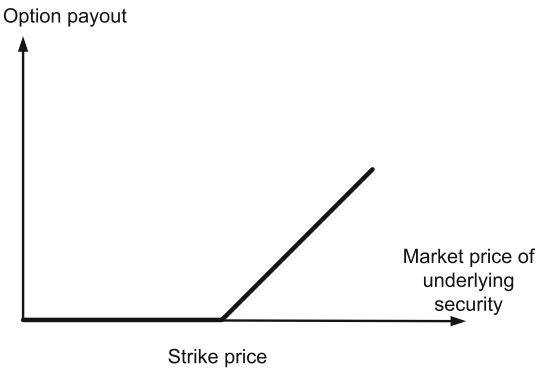


Fig. 3.3 Value of call option as a function of the price of the underlying security

using multiple technologies available to the decision-maker, but which may not be available to investors.

Investors use options to implement strategies for portfolio management. In the same way, decision-makers may invest in recourse capabilities to improve their capabilities to manage uncertainty. In a sense a stochastic program with recourse can be viewed as an option portfolio selection model. However, recourse is a concept that goes far beyond options. Recourse is modeled from the collection of possible actions and resources available to the decision-maker—so in a sense, the use of recourse models allows decision-makers to design their own options.

3.3.1 Multiple Outcomes

Typically in stochastic programming, you are concerned about multiple outcomes. In the example above, we were concerned about maximizing mean return and at the same time wanted to minimize the shortfall measure. The natural thing to do is to parametrize the problem

$$\max_{x \in X} \left\{ \sum_{s \in \mathcal{S}} p^s x^s - \lambda \sum_{s \in \mathcal{S}} p^s h(v - x^s) \right\} \tag{3.6}$$

Varying the parameter gives an "efficient frontier" of solutions; $\lambda = 0$ gives the solution that maximizes expected return, and as λ grows higher and higher, it will give the solution that minimizes the shortfall measure.

This "multi-objective" style of optimization procedure is very common in stochastic programming. It allows us to describe multiple targets and objectives, and the optimization process generates efficient sets of solutions each with different properties relative to the targets.

3.4 Expected Utility

The problem of choosing outcome probability distributions has a very deep technical literature that centers around the economic utility theory developed by Von Neumann and Morgenstern [84]. The basic idea is that preferences between outcomes distributions (following certain rules) can be modeled by choosing an outcomes distribution that maximizes *expected utility*, where utility is modeled by a concave function of the outcomes. We will give just a brief outline here of the portfolio selection problem in finance, since it is in finance that the basic assumptions of expected utility are likely to be satisfied.

Let us place ourselves in the realistic world of choosing to invest in corporations that are in effect managing portfolios of businesses. As observed above, one can anticipate some statistical regularity of outcomes, which will be observed as dividend payments or changes in the market prices of company stock. Purchasing a single share of stock in company i will produce an annual return of $r_i(s)$, where s is a scenario parameter indicating the strength of the market returns modified by the idiosyncratic performance of management. (Just to be clear on the meaning of return, we will adopt the convention that a return less than 1.0 represents a loss and greater than 1.0 a gain.)

Investing in a portfolio $x = (x_i, i \in I)$ of companies will therefore produce an outcomes distribution

$$s \mapsto \sum_{i \in I} x_i r_i(s), \text{ with probability } p(s). \tag{3.7}$$

Since companies are run by managers who are rewarded for "earnings growth," it is quite likely that there is some statistical regularity to the dividend payments. Under the assumptions of *expected utility*, then there exists a utility function $F(\cdot)$ such that the optimal choice of outcomes distribution is given by maximizing expected utility

$$\max_x \sum_{s \in \mathcal{S}} p^s F\left(\sum_{i \in I} x_i r_i(s) \right) \tag{3.8}$$

Now we ask the question—what should the utility function be? In the case of expected value optimization, the utility function is just the identity: $F(R) = R$. Should we use the expected value criterion to choose an optimal collection of stocks? What are the other choices?

If our perspective is very long term (for the rest of our long lives, for example) and our objective is to simply take the money every year and *spend it*, then the expected value discussion applies: The variability over many years will oscillate around the mean.

On the other hand, if we take the money and *reinvest it*, then the story is different. When the returns are identically distributed, the strategy that achieves the maximum wealth is the one that *maximizes the logarithm* of the return. (Of course, this is simply the mean of the exponential growth one achieves through reinvestment—so the mean wins out here too!). This result is originally due to Kelly and has been developed in the stochastic programming context by Ziemba and his colleagues [61].

Of course, these objectives assume we know the distributions rather precisely. In fact, the distributional characteristics of investment returns change over time. The next section investigates an important tool used by portfolio managers to model the risks of investment.

3.4.1 Markowitz Mean–Variance Efficient Frontier

You have likely heard of the Markowitz criterion, in which the objective minimizes the variance for a given level of expected return [64]. Why is this so popular? Well, for one thing it uses observable statistics—mean and variance of financial returns are easily observable. But is it sensible? After all, variance penalizes both downside risk and upside risk. Is it reasonable for an investor to choose a criterion that minimizes the risk of going higher?

To answer this question, let us consider an investor who knows her statistics. For instance she knows the mean return vector m and the variance–covariance matrix V. She also knows her Von Neumann–Morgenstern and wants to choose her portfolio according to maximum utility. But what utility function? She goes to a website that offers to discover her utility by asking questions about one gamble after another. But she is really not sure about this at all. So many comparisons!

Along comes a slick stochastic programmer who offers to give her an *entire collection of optimal portfolios* to choose from. Every one will be optimal for some utility function. And up to a second-order approximation, every utility function will be represented. How does he do it?

He argues like this. Suppose your utility function was $F()$, and we know how to find its optimal portfolio, namely the $\hat{x}$ that maximizes (3.8). Then of course we know its expected return, namely $\hat{R} = \sum_i m_i \hat{x}_i$. Expand the utility function to second order around this expected return:

$$F(R) \sim F(\hat{R}) + F'(\hat{R})(R - \hat{R}) + 1/2F''(\hat{R})(R - \hat{R})^2 \tag{3.9}$$

Now find the maximum utility using the right side of the approximation instead of the left:

$$\max \sum_s p^s \left[F(\hat{R}) + F'(\hat{R}) \left(\sum_i r_i(s) x_i - \hat{R} \right) + 1/2F''(\hat{R}) \left(\sum_i r_i(s) x_i - \hat{R} \right)^2 \right] \tag{3.10}$$

Without loss of generality, let us also restrict the search to those choices that satisfy

$$\hat{R} = \sum_i m_i x_i \tag{3.11}$$

Notice that this does not specify the choice of x (it does narrow it down considerably, but let us keep going). The main point to keep in mind in the argument is that the choice of F determines $\hat{R}$. Now with this narrowing down, let us look carefully at the approximate utility maximization. First, note that the term $F(\hat{R})$ is fixed, so it will be ignored in the approximate maximization. Second, note that the second term disappears! This is because we are restricting our choices of x to those that lie on the mean hyperplane (3.11). Finally, note that the last term consists

of the second derivative of a concave function (which is negative) multiplied by the variance of the return. When we clear the negative term from the objective, the maximization turns to a minimization, and we are left with the problem of minimizing variance subject to a constraint on the mean return.

It follows that the approximation is none other than a version of the mean–variance problem central to the Markowitz method:

$$\begin{array}{ll} \min & \sum_{i,j} x_i V_{ij} x_j \\ \text{such that} & \hat{R} = \sum_i m_i x_i \end{array} \tag{3.12}$$

As we vary our choices of utility $F()$, we will also vary our choices of return $\hat{R}$. It follows that all our choices will lie on the *efficient frontier* of solutions that minimizes variance for a given level of mean return. This is the approach originally formulated by Markowitz [64]. The mean–variance efficient frontier does in fact present our investor with a collection of points that a utility maximizing investor would choose, up to a second-order approximation.

How good is the approximation? Well, this is something you can try for yourself. Find some tables of annual returns of large corporations over the past 20 years, calculate the means and variances, and answer for yourself: How good is the second-order approximation to your favorite utility function, say, the logarithm? You will find that it is pretty close. After developing this argument (in [53]), we asked this question. For the logarithm function, it seemed like the second-order approximation was very sensible for absolute returns in the range of 75% to 400%—which is one very good reason for the popularity of the Markowitz method over the 60 years since its discovery.

The other reasons are that the parameters are quite easily observed in the marketplace. The covariance matrix and mean can be constructed by observing a time series of annual returns. The sequence of observations are viewed as samples drawn from the distribution of future returns. Standard statistical calculations can be used to provide appropriate estimates. It appears that there are long-term cycles in *market* volatility; however, the same cannot be said about variances and correlation terms for individual stocks—partly because there is less data to estimate them, and partly because the relative performance of company stock prices depends on so many factors. However, the actual performance of the mean–variance model over time is much more sensitive to the estimation error in the fundamental parameters, most especially the mean.

At this point we shall close this discussion. This is not a book about statistical arbitrage in financial markets, it is a book about modeling choices under uncertainty. We hope we have conveyed to you some of the flavor and language of expected utility and the mean–variance approach as it is applied in investing.

The interested reader can go much further, of course. However, in the end, we would like you to be aware that many market practitioners, on the basis of much experience, do not believe that past data inform future behavior of prices. Rather, the basic reason for prices to move one way or another is due to supply and demand—

and in the securities market the dynamics of supply and demand are affected at times by overwhelming optimism and at times by overwhelming pessimism and always amplified by leverage. This brings us to our next topic.

3.5 Extreme Events

> The 50-50-90 rule: Anytime you have a 50-50 chance of getting something right, there's a 90% probability you'll get it wrong.
> – *Andy Rooney*

Sometimes extremely bad things happen. Asteroids strike, virulent diseases break out, markets crash, and products fail. Models with uncertainty must consider the consequences of extreme events.

In the financial industry, for example, regulators require some institutions to estimate the upper tail of the loss distribution and to hold reserves proportional to these loss estimates. The intention of the regulators is to force the industry to model the worst-case extreme losses of their portfolios and to penalize extremely risky positions by forcing them to hold safe, but low-yielding securities, in proportion to the extreme loss potential. The upper quantile of potential losses is called *value at risk* or VaR. A related, increasingly popular, and in our view preferable, statistic is called the *conditional value at risk* or CVaR—which is the expected value of the losses that fall above the VaR. This is illustrated in Fig. 3.4.

The first step in analyzing extreme risks is to decide what to analyze. In our investment problem above, we could decide to set a *benchmark* return. For example, we could choose a level R_s of returns (keep in mind that a return below 1.0 means that our investment has lost value) that represent an absolute threshold below which we do not want to go. Then our *losses* relative to the benchmark are

$$L^s(x) = R_s - \sum_i x_i r_i^s \quad \text{for } s \in \mathcal{S} \text{ with probability } p_s \tag{3.13}$$

The VaR for the losses L corresponding to a quantile Q is calculated by sorting the losses from lowest to highest and counting from the lowest loss up until you have

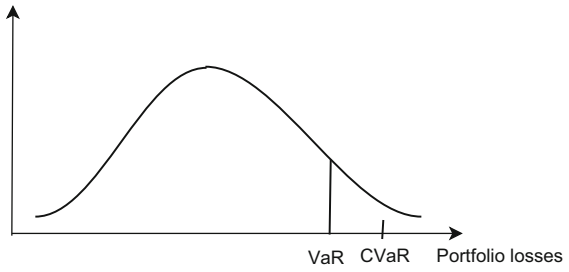


Fig. 3.4 Density function, VaR, and CVaR for potential losses. VaR corresponds to a chosen quantile, while CVaR is the expected loss above VaR

counted a proportion $Q\%$ of all scenarios. Here is a formula that says the same thing but is switched around to highlight something that will become important a bit later on

$$\text{VaR}(x;\, Q) = \inf_{L} \text{ such that } \#\left\{ s \in \mathcal{S} \text{ with } L^s(x) - L \geq 0 \right\} \geq \frac{1-Q}{100}, \tag{3.14}$$

where the "number sign" # in front of the set indicates the number of points in the set. The risk level $\text{VaR}(x;\, Q)$ divides the sample space of losses into two parts: good and bad. The good points are the Q% that have losses below $\text{VaR}(x;\, Q)$, and the bad points are the (100-Q)% that have losses greater than the $\text{VaR}(x;\, Q)$.

In banking and finance one says "our 99% VaR is USD 0.5B," and it is understood by bank regulators that the probability that the bank's position will suffer a loss greater than 500 million dollars is only 1%. These statements are taken quite seriously. In fact, the bank must demonstrate to the regulators, by reconstructing their positions backward in time, that their past 200 daily 99% VaR loss estimates were violated no more than twice.

But can you see that VaR behaves a bit strangely as a function of x? Try this thought experiment: Take the position x_i in the security with the absolute worst losses and make a small increase in it. A good risk measure should increase when you do more bad things. Is the VaR guaranteed to change? No, it is not, and this is why it is a controversial measure.

The CVaR is defined to be the expected value over the quantile in the definition (3.14) of VaR. Given that expected values are linear, one would anticipate that a risk measure based on CVaR will behave as a risk measure should. However, there is the complicating matter of locating the support of this expected value, which behaves badly as we have seen. A relatively new result by Rockafellar and Uryasev shows that CVaR is indeed a well-behaved risk measure. They proved that its value is given by the following stochastic program:

$$\text{CVaR}(x;\, Q) = \inf_{L} \left\{ L + \frac{1}{1-Q} \sum_{s} p^s \max[0,\, L^s(x) - L] \right\}. \tag{3.15}$$

Do you see that this calculates expected value over the same set that we counted points over for the calculation of VaR?

In usage, VaR and CVaR have different purposes. Both are measures of the tail behavior of the outcomes distribution. VaR tells you something about where the tail is supported, and CVaR gives you more information about how the distribution behaves over the tail.

Regulators prefer VaR, because they have more confidence in the statement "Only 5% of losses are greater than VaR" than the statement "the average of the worst 5% of losses is CVaR." The second statement requires a model of the tail distribution, which by necessity must always be based on a sample of very few observations. People can disagree about tail distributions, but the definition of the

location of the tail is usually pretty well estimated. Regulators will prefer to stick with estimates for which there is a broad agreement.

On the other hand, decision-makers should prefer CVaR, for two reasons. First, precisely because it does require a model of the tail distribution, the inclusion of CVaR will force the decision-makers to think about how bad things can get.

We can think of CVaR as being just like a target shortfall measurement, but where the target is specified in terms of the quantile instead of some arbitrary target. Of course we could just specify the target to be high enough to be "in the tail." The advantage of using CVaR is just the fact that we do not have to guess beforehand where these tail values are.

Sometimes it is a good idea to use a different distribution for extreme events. For example on trading floors, the traders do not use the same distributions as the risk managers. Risk managers have different objectives and different time horizons. A trader about to make a quick killing should not be concerned about surviving a low-probability market crash over the next 30 days. It is the risk manager's job to worry about that. The risk manager does not do this by influencing the trader's models. (It is hard to imagine a trader accepting any kind of modeling advice from a risk manager!) Risk managers do their job by limiting trader access to the firm's capital. It is entirely reasonable that the risk manager and the trader use different distributions to achieve their purposes.

Extreme event modeling in finance is gradually being extended by regulators to cover the so-called operational risks. This covers things like power failures at data centers, losses due to failures of internal controls, rogue traders (and risk managers), and so forth. The financial system is so inter-linked, and the traded volumes are so great that an operation breakdown, say, due to a power failure at a major bank's data center, could have widespread financial and economic consequences for weeks afterward.

Companies in industrial sectors with large extended supply chains are also beginning to model the tail distributions of bad events, operational and otherwise. Examples of bad events could be: the bankruptcy of a key supplier or creditor, quality control failures internally or of key suppliers, a major product liability lawsuit, improper release of private information of consumers, earthquake damage to a key electric power supplier, as well as losses in financial portfolios that back employee insurance programs or working capital.

3.6 Options

This section is for readers who have some experience with financial options (and for those who are curious!) and who may wish to understand how options pricing fits into the stochastic programming framework.

This section presents a quick guide to options theory from the perspective of stochastic programming and then discusses how options fit into stochastic

programming models. The background for this brief overview can be found in King [55], King, Streltchenko and Yesha [56], and King, Pennanen and Koivu [54].

Consider the problem of transporting goods to markets. There are two types of risks. One comes from the demand: how much to deliver and when to deliver it. The second comes from the cost of delivery. It is quite likely that there will be competition for transportation services in these markets. Transportation costs can be quite significant, and the unit prices can be quite variable and seasonal.

Transportation services are mostly standardized, and there exist well-developed markets in shipping contracts. It is quite possible to purchase *forward contracts* or *futures contracts* to ship containers of a certain dimension between major hubs, like Shanghai and Los Angeles. Of course, there must be some relationships between these prices. For example, the Hong Kong to New York price must be related to the Hong Kong to Los Angeles and the Los Angeles to New York prices.

There are three concepts underlying the operation of these markets:

- Replication: Contracts with promises to deliver a good or a service at future dates can be replicated by a trading strategy in the related "spot" markets for the good or service.
- Arbitrage: When there are discrepancies between prices in related markets, then speculators enter the market and try to make money from the price differences.
- Liquidity: Small gaps between the prices of the highest bid to buy (best bid) and lowest offer to sell (best offer) make it easier to execute trading strategies.

Liquid markets are also called efficient markets in the literature. Another use of the term liquidity is to indicate the flow of money in a market: liquidity moving in to a market indicates more buyers than sellers, and liquidity moving out means more sellers than buyers.

3.6.1 A Simple Options Pricing Example

Consider a very simple example. Our manufacturer wants to decide whether to purchase a contract today for $8000 that promises to ship 100 containers in 1 year's time, or an option offered by a trader today that promises to deliver the contract for a payment of $8000 a year from now.

Now suppose the manufacturer is experienced enough to know, historically, that there are really only two interesting alternatives for the spot price in 1 year's time:

- The spot can either go up to $10,000 (because the ships are full).
- Or it can go down down to $7000 (because the ships are empty).

This looks like a stochastic program! By this point in the book, we know what to do:

1. Develop a model that incorporates the futures contract into the manufacturer's planning problem.

2. Assign probabilities to the scenarios for the future spot prices.
3. Create a risk measure for the decision problem.

But now let us discuss how the trader thinks. The trader develops a replication trading strategy. In the first stage, the trader receives the money from the manufacturer, borrows some more money, and buys some quantity of the future contract. In the second stage, the trader sells the future contract bought in the first stage and pays back the loan. This is a stochastic program! Can you write it down?

The solution depends only on the three numbers: \$7000, \$8000, and \$10000. Here it is:

- First stage:]
  - Trader sells the option to the manufacturer for \$666.67.
  - Trader borrows \$4666.66.
  - Trader purchases 2/3 forward contracts (which costs $2/3 \times 8000 = 4666.66 + 666.67$).
- Second stage:
  - Spot goes up to \$10000.
    - Manufacturer wants to "exercise" the option.
    - Trader sells the 2/3 forward contract for \$6666.67.
    - Trader pays back the loan of \$4666.66.
    - Trader gives the manufacturer \$2000.
  - Spot goes down to \$7000.
    - Manufacturer does not "exercise" the option.
    - Trader sells the 2/3 forward contract for \$4666.66.
    - Trader pays back the loan of \$4666.66.

This model illustrates the market requirements for this kind of financial wizardry:

- Replication: The trader's model has an optimal feasible solution (this is a theorem).
- Arbitrage: The price of the option will be set so that the trader is left with exactly \$0.00 along each scenario—so the option price is "risk-free."
- Liquidity: The trader can buy or sell at the prices required by the model solution.

Now let us discuss the theorem. The replication problem has a feasible primal solution—we just showed that. To prove that it is optimal, we formulate the dual problem, show that it is feasible, and apply the linear programming duality theorem.

The dual problem is

$$\begin{aligned} \max_{q \geq 0} \quad & 2000q^1 + 0q^2 \\ \text{such that} \quad & \begin{cases} q^1 + q^2 & = 1 \\ 10{,}000q^1 + 7000q^2 & = 8000 \\ q^1, q^2 & \geq 0 \end{cases} \end{aligned} \tag{3.16}$$

Compare this with your formulation of the replication problem. Is this dual problem correct?

Assuming this is correct (it is), let us interpret it. The first and last constraints in the dual problem show that the q's are the weights of a probability distribution because they are non-negative and add to one. The second condition is a kind of integration constraint, which says that the expected value of the future spot prices using the probability distribution q equals the price today. The technical term for a distribution that satisfies a condition like this is *martingale*.

The probability weights of the martingale have to satisfy

$$10{,}000q^1 + 7000q^2 = 8000.$$

There is *exactly one* answer: $[\frac{1}{3}, \frac{2}{3}]$. For this answer the value of the dual objective is

$$2000\frac{1}{3} + 0\frac{2}{3} = 666.67$$

Since there is only one answer, we conclude that \$666.67 is the correct price for the option.

This simple options pricing model demonstrates a number of important points for interpreting financial market information for modeling stochastic programs.

Martingales
The existence of the Martingale is the key step in the options pricing model. In our simple model, a martingale exists if and only if there are scenarios with future prices above and below the current price. We made sure of that, and that is why the model worked.

Calibration
Martingales have parameters. In the case of our simple model, the parameters are the spot price \$8000 and the fact that prices either go up by \$2000 or down by \$1000. The option price of \$666.67 actually *implies* these parameters! Options prices are used in practice to *calibrate* the Martingale parameters. Calibrating a Martingale that is consistent with all options prices is generally impossible (see [54]). Details matter: timing of dividend or interest payments, the number of trading days until expiration, carrying costs, opportunity costs, and so forth. The basic approach is to use prices of liquid options to calibrate and then apply this Martingale to estimate prices for the less liquid options.

3.6.2 The Hidden Risk of Options

Option prices change. If our manufacturer bought the option yesterday, then by today the "risk-free" price will already have changed. Our trader will already have made trades to hedge those changes (that is the job of the trader). The reason is that the *Martingale* has to be re-calibrated.

Although the option price yesterday is called "risk-free" or "risk-neutral," this is only true in the context of yesterday's Martingale. The Martingale was calibrated to yesterday's prices. As we noted, calibrating Martingales is a complex process that has to take into account all the liquid instruments, all the details of these instruments, and not only options that mature in 1 year but also all those that mature before and after. In large institutions, which are the ones that provide liquidity to the market (market makers), this is the job of specialists. The trader's job is to deal with all that complexity, as well as to develop an understanding of the daily, weekly, and annual cycles of supply and demand, as well as the behavior of the other traders in the market. These are the *real risks* the trader faces.

It is possible to create a model for the trader problem. It is an inherently multistage problem. But trader performance is difficult to pin down. Generally, the trader is given a big job, for example, to trade a large number of options and other instruments bought by all manufacturers (a block). Some blocks are easier to trade than others. Why should a trader suffer if the assignment is to trade in a crazy market like container futures? As is always true in the modeling business, the obvious objective (profit)—while important overall—has to be adjusted to match the context.

The largest risk faced by traders is due to sudden and very large changes in market liquidity. In some cases, the bid and offer price gaps are so large, even infinite, and prices are unreliable. Unreliable prices lead to knock-on effects in the market. For example, traders may not be able to borrow because the trader's portfolio cannot be priced.

When this happens, the trader goes bankrupt, the manufacturer's option is at risk, and suddenly there is a risk to the real (as opposed to financial) supply chains. At this point, the central banks become the "lender of last resort" and find ways to replace the missing liquidity.

We make this comment about the extreme case because it illustrates the main role of the trader: The trader provides liquidity. The first activity of our trader is to borrow money. That money does not come from thin air, of course, but this feature of the option is not visible to the manufacturer. This is sometimes called "hidden liquidity." The management of this hidden liquidity is the real problem of the trader.

3.6.3 Financial Options in Stochastic Programming Models

Our manufacturer has to plan for many risks. Let us say that the main one is demand, and the manufacturer has constructed a scenario tree for future demand. As demand

changes, the manufacturer requires more or less transportation services. How should the manufacturer incorporate these effects into the planning model?

Transportation services is a liquid market. Liquid markets offer many types of contracts that are priced with calibrated Martingales, and the calibration data are *always today's data*. In the future, represented by the demand scenario tree, there will be different Martingales calibrated to the future data.

The option offered by the trader may be valuable to the manufacturer. To find out how valuable, the manufacturer can model how much transportation is required in each demand scenario. Then the manufacturer can create first-stage decision variables:

- How many future contracts to buy today?
- How many options on futures contracts to buy today?

Can you see that this will require that the manufacturer extend the future demand scenario tree to include the future spot prices of the contracts?

The choice faced by the manufacturer is whether to include the future spot prices in the scenario tree. This is a judgment call. There are additional model risks: the correlation of demand with transportation costs, how the larger model can be solved, and then how to interpret the results. The manufacturer may or may not be comfortable with the additional complexity. Sometimes it is better to let the traders do the work. At least at this point you will understand a bit more about what traders do and what sorts of risks they cope with.

3.7 Learning and Luck

> I'm a great believer in luck and I find the harder I work, the more I have of it.
> – *Thomas Jefferson*

Learning is a subject of its own, and we shall not have a general discussion of the issue here. However, since phrases like "learning organizations" and "organizational memory" are pitched by many consultants, we would like to mention but one issue which is relevant for this book. Learning presumably implies that from the outcome of a decision we wish to become wiser. Next time we make a similar decision we wish to either make a better decision (if this one was not so good) or an equally good one (if this one was good). For this to make any sense, there must be a causal relationship between what we did and what happened.

If you walked backward into the shop where you bought the Lotto coupon on which you won a lot, you may think (many certainly do) that this is good for winning, and you may choose to do the same in the future. We all know that this is nonsense, and we call it superstition, not learning. If instead you have a model you use for making some decision, and the decision leads to, say, a very good outcome, do you then know it was a good decision and a good model? Is there a logical connection between the ex post observation and what you did? Let us again

exaggerate a bit. Assume there are two gambles, both with the same price for taking part, and both with the same prize if you win. Assume in one gamble there is a 10% chance of winning the prize, in the other 90%. Assume you play the game with 10% chance and, voila, you win! Does that mean you made a good decision? Of course not. Is there anything to learn from the ex post observation? No, there is not. You were simply stupid and lucky. We can say that because there is an ex ante evaluation that in this simple case tells us that what you did was stupid. The fact that you were lucky does not change that.

So what is learning? We shall not answer that. But be aware that learning is a difficult issue in a random environment. In genuine decision contexts, there is a causal but stochastic relationship between what you do and the consequences. But to learn can be hard, because you must separate luck from cleverness.

But do we care? Maybe for our own decisions we often do not. We prefer to value our decisions based on ex ante analysis, looking for decisions that according to our own utility function maximizes expected utility. But there is a context where we really are interested. If you are to engage a consultant, he probably sells himself based on his track record (sounds great does it not?) And now we are interested in ex post learning. Is he really good, or are we just facing a lucky guy? Could it be that he looks so good because he takes too many chances, that this is definitely not the one we want? Some say that if your broker makes a lot of money for you over a reasonable period of time—fire him! He takes too big risks.

Chapter 4
Scenario Tree Generation

With Michal Kaut and Jamie Fairbrother

> It is a very sad thing that nowadays there is so little useless information.
> – *Oscar Wilde*

> Weather forecast for tonight: dark.
> – *George Carlin*

So far we have talked about models and structures with respect to uncertainty. But we have not talked about data. We have talked about prices and demand, deterministic or stochastic, and we have talked about distributions of random variables. But rarely are these available in the format we need for our algorithm.

We shall not bring up most questions of data and data manipulations and estimations here. Much of that belongs elsewhere, in your course in statistics, econometrics, or maybe forecasting. Maybe your data come from experts, or maybe all you have is your own humble understanding of the random phenomena governing your model. Whatever you have, this book picks up the modeling process at the point where you have all the data you are able (or willing) to collect. So you may have historical data, possibly vast series of them, and you may have actual distributions, estimated, or assumed.

In order to find a solution to your stochastic program, you will most likely need to create a *discrete version* of your probability distribution. Many algorithms used to solve decision problems under uncertainty require a discrete distribution, especially algorithms designed for general purpose problems. (Some problems are designed, with specialized distributions and compatible formulations, to be solved "in closed form" without a discretization step.)

It is important to realize from the very beginning that the resulting discrete distribution is not part of the actual problem under investigation, but something you might need *because of the algorithm you have chosen*. When investigating your problem, your interest is most likely the random variables themselves (dependent/independent, normal/log normal/uniform, ...) and of course the algebraic

A. J. King, S. W. Wallace, *Modeling with Stochastic Programming*, Springer Series in Operations Research and Financial Engineering,
https://doi.org/10.1007/978-3-031-54550-4_4

formulation of the model. But in most cases such a model cannot be solved as a stochastic program. Hence, you need to discretize. However, since there is no obvious way to pass from random variables to a (good) discretization, we choose to see the discretization as part of the modeling. Different people will discretize the same data in different ways, and hence, we can see the discretization as a way to model the random variables. Of course, it is also valid to view the discretization as part of the solution procedure. We have not chosen that view, however, mostly because for most algorithms used to solve stochastic programming problems, there is no obvious way to go about the discretization, and hence, we wish to point out that the process must be governed by the modeler, that is, you.

It is your responsibility to make sure that random variables are represented in an appropriate way. There is no general way that always works.

Because the discretization is not part of the problem, it may not interest you from an academic point of view. But how you do the discretization is actually crucial for your results. Hence, this chapter has two purposes, one is to explain *why* this part of your modeling process is crucial, and another how you can try to go about the discretization at minimum effort if this is not your real area of interest.

At the simplest level, the goal of the discretization procedure is to make sure that it is not the discretization procedure itself that drives your optimization problem, but rather your model—meaning your algebraic formulation and your model of the random variables. In a sense, the goal is to totally hide the discretization procedure so that things are *as if* you had solved the model with the original distributions. If there are users of your model, those users are most likely interested in the algebraic model (or at least what it represents) and may be interested in discussing the random variables as such. But they will hardly ever be interested in how you discretized the distributions. They will tell you that "that's your problem."

A good discretization procedure cannot be observed in the solution to the model.

Let us be a bit more specific about why this is important. Assume that you, for example, have a stochastic network design problem, and you wish to find out how the optimal design depends on the expected demand between nodes a and b. Hence, you solve your stochastic program several times, varying the expected demand over the relevant area. You obtain a certain set of solutions, and you study the solutions to understand how the design depends on the expected demand. But your discretization procedure also depends on the expected demand. As the expected demand varies, you will be producing different discretizations, and these are used

in the optimizations. Your concern now ought to be: Are the observed changes in optimal design a result of the discretization, or are they actual changes in the optimal design? Are the observed changes real, or are they systematic or random effects of how you discretized? Are you sure you now know how the optimal design depends on the expected demand between nodes a and b? Or is everything just noise from your discretization? The purpose of this chapter is to help you ensure that it is not the discretization that drives the results.

The premise in this chapter is that you have chosen a solution algorithm for your stochastic program that simply requires a scenario tree as input. However, there are some cases where this is not needed.

First, there are special *solution algorithms* that have sampling included as a part of the solution procedure. One example is stochastic decomposition, as defined by Higle and Sen [38, 39]. Also this algorithm uses scenario trees, but they are sampled as you go along. The algorithm does not suffer from the difficulties that will be mentioned in Sect. 4.1.1. Be aware that the algorithm is very challenging to implement due to its dependence on efficient data structures and that it can be used only for stochastic *linear* programs.

There are two other such methods that work for convex, not only linear, problems: stochastic quasigradients by Ermoliev and Gaivoronski [21, 27], and importance sampling within Benders decomposition by Dantzig and Infanger [14, 44].

There are also special cases, such as versions of the newsboy problem, where specialized algorithms have been developed for special cases, such as normal distributions, and where discretizations are not needed.

4.1 Creating Scenario Trees

We shall, as we mostly do in this book, present the theory in a two-stage setting. Hence, there will be no time index on the random variables. In multistage problems, discrete random variables will, due to dependence, take the form of a tree, where the root represents "today" and the tree branches for each stage. In the two-stage case, the tree is simply a "bush," with one node for "today" and one node for each possible "tomorrow." Even so, we shall allow ourselves to talk about scenario trees also in that case.

The goal is now to create a scenario tree such that the error caused by discretization is as small as possible, while maintaining solvability of the model.

4.1.1 Plain Sampling

An obvious way to create a scenario tree is to sample from the underlying distribution. If you do this, eventually, you will end up with a tree (discrete

distribution) that is arbitrarily close to the distribution you started out with. So why is this not the whole story? Why not simply sample a "large enough" tree, and then proceed? The answer is simple enough. For all but the simplest problems, you will be left in one out of two situations:

1. Your tree is so large that the discretization error is acceptably small. But it is also so large that you cannot solve the optimization problem. You have a numerical problem.
2. The tree is so small that you can solve the optimization problem, but also so small that the discretization is a bad representation of the underlying distribution. You have a representation problem.

There is a theoretical basis for this statement as well as practical experience. Shapiro [79, 80] shows that

$$N \geq \frac{12\,\sigma_{\max}^2}{(\epsilon-\delta)^2}\left(n\,\log\frac{2\,D\,L}{\epsilon-\delta}-\log\alpha\right)$$

which "shows that the sample size which is required to solve the true problem with probability $1-\alpha$ and accuracy $\epsilon > 0$ by solving the SAA problem with accuracy $\delta < \epsilon$, grows linearly in dimension n of the decision vector x" [79, pp. 375]. Here, D is the diameter of the feasible region, L is the Lipschitz constant of the objective, and $\sigma_{\max}^2$ is a bound on variance of a function of the objective, used to derive the result.

This might look daunting, and indeed it is. If you like to try and put in reasonable numbers, you will see that N has to be extremely high for reasonably sized problems. So if you wish to guarantee a reasonable quality solution for a reasonably large problem, you are indeed in trouble.

Sampling might of course in many cases lead to much better solutions than indicated by this inequality. But to realize that that has happened you need other tools—tools that test the quality of a given solution. We shall return to that later.

So, there is an important point here. A procedure that has all the required properties in the limit may not have any desired properties if not taken to the limit. A tree made by a procedure that has perfect limit properties may still be arbitrarily bad for its purpose.

> Although sampling has perfect limit properties, there is no guarantee that a small sampled tree has any useful properties when used in a stochastic programming model.

Sampling can be improved by correcting means and variances ex post by using a simple linear transformation on each margin. Since many stochastic programs are sensitive to expected values [see, for example, 51], this can improve the scenarios

dramatically. Additionally, one can employ variance reduction techniques such as stratified sampling, or antithetic sampling [41]. These are sampling techniques that, rather than generate independent samples from the distribution in question, aim to generate samples that result in lower variance in the sample average of a random variable. These might work, for example, by sampling scenarios more evenly across the support of a distribution.

Another, possibly less obvious issue, is that to sample you need to know what you should sample from. Often, you do not fully know your distribution. To sample, you might need to add information (that possibly is arbitrary) to have something to sample from. Some of the ideas to be presented in this chapter may not need the full distribution. In some cases that is an advantage.

4.1.2 Empirical Distribution

If all you have is history, and you have no qualified reason to believe that the future will be different from the past, then what do you do? In particular, do you use the data (history) to estimate a continuous distribution as part of your modeling, or is that not a good idea? Remember that, eventually, you are going to revert to a discrete distribution.

If you use the data to estimate some parameters of a distribution, you are adding information to the data. If this information is arbitrary, you are adding meaningless information. That is not a wise thing to do. On the other hand, if you know, or have reasons to believe, that the data come from a specific family of distributions, estimating that distribution is a way to add your extra information.

So, the story is: If you have nothing but historical data, use that data as your distribution. So when we in what follows refer to a distribution that needs to be discretized, we might be referring to the empirical distribution, as it might have too many outcomes for the stochastic program to solve in reasonable time.

You may have your own ideas about how to make a scenario tree. If so, you should certainly investigate them. But do not use them blindly. Rather, use the methods that follow in this chapter to test your methods to see if they actually behave well. Otherwise, you may find yourself in trouble later on.

4.1.3 What Is a Good Discretization?

It is worthwhile thinking for just a short while about we should mean by a good discretization. We already mentioned that it should be one that does not affect the solution to your problem. It should function in such a way that it is as if you had used the original distribution. Consider the two discrete distributions in Fig. 4.1.

In the figure, we have also drawn the density of the standard normal distribution $N(0, 1)$ for comparison. Would you say the discrete distributions are good

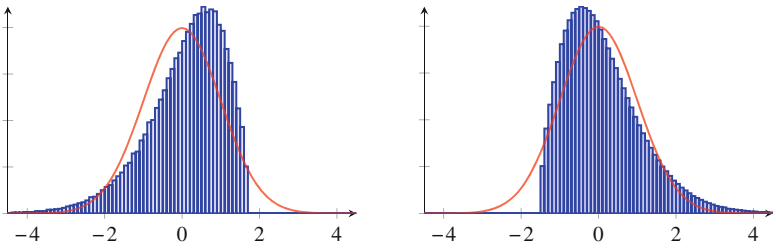


Fig. 4.1 Histograms of two discrete distributions, compared to the density of a normal distribution with the same mean and variance ($N(0, 1)$)

approximations of the normal distribution? And similarly, would you say the left discretization is a good approximation of the right one?

Well, that is in fact hard to answer, and the reason is at the core of the problem of discretizations: It depends on the use! The Markowitz model for portfolio management (see Sect. 3.4.1) will give the same solution with these three distributions. In fact, in any dimension, the model only reacts to the first two moments (including cross moments) of the distribution, and the three distributions in the figure have the same first two moments. So despite the fact that they look different, and in fact are very different (in particular the third moments are very different), they each serve us equally well.

> The quality of a discretization depends on the model in which it is used.

Do you think this is obvious? Well, maybe it should be. But many approaches in scenario tree generation seek a discretization that makes the scenario tree as similar to the original distribution as possible, *without regard to the stochastic program needing them*. This has the advantage that generating scenario trees can be done without knowing the corresponding optimization model and in fact can be done by someone with pure statistics expertise. In such a setting, the three distributions in Fig. 4.1 would probably be declared rather different, and in particular, not good approximations of each other.

Now, if you find a scenario tree that is close to the original distribution in this purely statistical sense (statisticians have ways of measuring the distance between two distributions), and that the discretization leads to a solvable stochastic program, *and* the resulting model is stable in a way we shall soon describe, then all is fine. But if you have a reasonably large problem, you might be in trouble. You might find:

1. That the optimization problem is not solvable, or
2. That the solution you have found is no good, even if your discretization was as close as possible to the original distribution, given the number of scenarios you

requested . . . and if you increased the number of scenarios in the discretization to get better solutions, you found yourself in item 1 instead.

Is there a way out of this loop? And does this have anything to do with how you measure the quality of your discretization? The answer is "maybe" on the first question, and "yes" on the second.

The reason is simple enough. The more aspects/properties of a distribution you care about (moments, extreme values, co-variation . . .) the more scenarios you need in order to have a good discretization. But if some of these properties, such as the third moments in Fig. 4.1 in case of the Markowitz model, do not matter to the optimization problem, why do you care about them? Think about a multi-dimensional distribution as a map. If you want a discretized representation of the map to look like the map itself, you may need many scenarios. But why care about parts of the map where you will not go? That is, why care about parts of the distribution that the optimization problem does not care about?

So, by only caring about the parts of the distribution that matters to your problem, you can get away with fewer scenarios in your discretization *or* obtain a better solution to your problem (i.e., get a better discretization) for a given number of scenarios, if you let your optimization problem guide you.

Apart from simple cases like the Markowitz model, where the results can be established analytically, we are not implying that it is easy to find out what properties of a distribution are important for your case. So far we have only established that by *thinking* that way, you may indeed bring a problem from being not solvable to being solvable. Just looking at the distributions, without reference to the optimization problem, may be easy, but may also be fatal. We shall return to this issue.

So let us turn to the next section where we ask: Is my procedure for generation scenario trees good? That is, do I have a good discretization? This section is particularly important if you have your own ideas about scenario tree generation. Make sure you do not use your procedure without testing it along these lines.

4.2 Stability Testing

So now we are back to our initial worry in this chapter: Can we be sure that when we test our decision model, we are not just testing the scenario generation procedure? Or more extremely, could it be that the results of the optimization model are not in any logical way a result of our algebraic model and random variables, but just random or systematic side effects of a bad scenario generation procedure? The ideas here stem from [49].

It is impossible to get all the way to the bottom of this problem. But we will see that we can get at least somewhere. We shall discuss two issues, *in-sample stability* and *out-of-sample stability*. The first represents a test of the internal consistency of a model (remember again, we view the generation of scenario trees as part of the model), and the second is related also to model quality.

We shall also discuss here the issues in terms of a two-stage model. Hence, a scenario tree is simply a bush, and we shall refer to it as $\mathcal{T}$. The problem at hand can thus be written as

$$\min_x f(x;\mathcal{T}), \tag{4.1}$$

where we have hidden the second-stage variables $y(\mathcal{T})$, and where it is implicit that we are to take an expectation or some risk measure with respect to the second argument, that is, the discrete distribution described by the tree $\mathcal{T}$. The true problem, the one we by assumption cannot solve, is given by

$$\min_x f(x;\boldsymbol{\xi}). \tag{4.2}$$

4.2.1 In-Sample Stability

In-sample stability does not have a deterministic counterpart. Assume you have a scenario generation procedure that is in itself random, that is, it does not produce the same scenario tree each time it is run with exactly the same data. This is typical of any sampling-based procedure, but true for many others as well. If your procedure generates the same tree all the time, there is a corresponding discussion that we shall return to below.

Assume now that you run your scenario generation procedure several times on the same data, producing many different trees. Let us denote them $\mathcal{T}_i$. Then run your optimization model equally many times, one time with each tree, that is, solve "$\min_x f(x;\mathcal{T}_i)$." Let the optimal solutions be given as $\hat{x}_i$. If the optimal objective function values are (about) the same in all cases, that is, if

$$f\left(\hat{x}_i;\mathcal{T}_i\right) \approx f\left(\hat{x}_j;\mathcal{T}_j\right)$$

you have in-sample stability. In-sample stability is in fact a totally instant-dependent property, but from a practical point of view, we shall assume that it is valid also for closely related instances. So, if you have in-sample stability, it does not matter which scenario tree you use, and hence, when solving your stochastic program, you simply take your algebraic model, your distributions, and then run first the scenario generation procedure, and then the optimization problem. And you do it, assured that the objective function value would be the same if you ran your problem again with the same data: What you do is internally consistent.

4.2.1.1 The Scenario Generation Procedure Is Deterministic

If you have a deterministic scenario generation procedure, we suggest you instead run your procedure repeatedly asking for trees of slightly different sizes. It is

unlikely that you have a deterministic procedure without any control over the size of the resulting tree. To have in-sample stability, you should observe (almost) the same optimal objective function value all the time. If you get totally different results just by adding, say, one scenario, clearly there is something wrong with your modeling.

4.2.1.2 Why Measure the Objective Function Value?

Stochastic programs tend to have flat objective functions. That is, in fact, one of the reasons for using stochastic programming. Hence, very different solutions can be more or less equal in terms of profit, cost, or whatever is the measure of goodness. As we do not want to declare it a problem to end up with a different solution if it serves us equally well as another one (after all, we are testing the scenario tree generation procedure), we choose to measure similarity by the objective function value. To achieve stability in terms of solutions is going to be very hard for problems with very flat objective functions. For example, in portfolio management, if there are two reasonably different portfolios that behave the same way, that is, which provide the same trade-off between risk and reward, we do not want to declare it a problem that they look different, at least not when the issue is the scenario tree. You are of course free to disagree with this view and require stability in solutions. Be aware, however, that that will be very hard to achieve.

4.2.2 Out-of-Sample Stability

In-sample stability was a test of the model itself. It represented a kind of robustness toward the discretization procedure. The test is totally void of issues of model quality. An in-sample stable model can be arbitrarily bad. Out-of-sample stability is also mainly an issue concerning the model itself but has aspects of something more in it.

Out-of-sample stability means that if you calculate the true objective function value corresponding to the solutions coming from different scenario trees, you get (about) the same value. In other words,

$$f\left(\hat{x}_i; \boldsymbol{\xi}\right) \approx f\left(\hat{x}_j; \boldsymbol{\xi}\right).$$

Hence, it is not such that your scenario generation procedure has generated a stability that is not really there. This could, for example, happen if your scenario generation procedure consistently avoided a difficult tail of the distribution, so that the in-sample stability you had observed was really just over a part of the support of the random variables. And the in-sample stable solutions you had found behaved very differently on the part of the support you had avoided. The true value showed you this, and you ended up out of sample unstable.

The starting point was that the true problem "$\min_x f(x;\boldsymbol{\xi})$" could not be solved. What we need to test in out-of-sample stability is slightly different, namely to fix x at $\hat{x}_i$ but then take expectations with respect to the true distribution. This is in many cases doable. If $\boldsymbol{\xi}$ is discrete (but with far too many scenarios to solve the corresponding true optimization problem), you may still be able to calculate $f\left(\hat{x}_i;\boldsymbol{\xi}\right)$ as that just amounts to solving a large number of second-stage problems. If $\boldsymbol{\xi}$ is continuous (or discrete but too large), the obvious method is to sample from the distribution in order to approximate the true value.

In many cases, the correct way to calculate the out-of-sample value is to construct a more detailed simulation model of your problem. That way, the out-of-sample calculations will not only take care of the fact that $\mathcal{T}$ is an approximation, but also the fact that f most likely only approximates the true problem as well. In a simulation model, you can normally add more details than in an optimization model.

You would also wish to have the objective function values coming from in- and out-of-sample stable solutions being almost the same. Otherwise, there is something going on you have not understood. Why are they different?

If we cannot evaluate $f\left(\hat{x}_i;\boldsymbol{\xi}\right)$, there is another, weaker, out-of-sample test we can perform: Let us have two scenario trees $\mathcal{T}_i$, $\mathcal{T}_j$ with corresponding optimal solutions $\hat{x}_i$, $\hat{x}_j$. If the model is out-of-sample stable, we should have

$$f\left(\hat{x}_i;\mathcal{T}_j\right) \approx f\left(\hat{x}_j;\mathcal{T}_i\right).$$

4.2.3 Bias

Even if you have both in- and out-of-sample stability, you may be in trouble. And this trouble is *bias*. You may be stable, but bad. This is not easy to test for in a simple way. You need to check if

$$\min_x f(x;\boldsymbol{\xi}) \approx \min_x f(x;\mathcal{T}_i), \quad \text{for all } i$$

Only if you can solve the true problem, can you make this test. Occasionally, your true distribution may be discrete, have very many scenarios, but even so, maybe you could solve the model once (using unreasonably much time) just to check.

Of course, if you have two scenario generation procedures for a given algebraic model, and both are in- and out-of-sample stable, but the true objective function values are different, then at least one of the generation procedures is biased.

To try to answer questions about quality other than stability, you will need to turn to statistical methods that we shall discuss a bit later.

4.2.4 Example: A Network Design Problem

Chapter 7 discusses a network design problem. In the model's early development, we ran into the problem of in-sample instability. This showed that something was wrong with the model (meaning the algebraic model plus the scenario generation procedure). The problem turned out to be that arbitrarily small changes in maximal demand could result in extra capacity being bought in the network. As capacity came in large pieces (trucks, in fact), these extremely small changes in demand resulted in big jumps in costs. Hence, no stability. Therefore, something had to be done. An in-sample unstable model is useless. What we did was to extend the scenario generation model to include the requirement that outcomes had to be within the defined support. Stability was achieved. This does not imply that we now had a good model. But at least the model now could be further tested. An in-sample unstable model cannot be properly tested, as it produces random results. For more details, see Chap. 7.

4.2.5 The Relationship Between In- and Out-of-Sample Stability

Despite the discussion above, there is not principally an ordering of stability results as we have indicated. It is, in fact, possible, at least in principle, to have out-of-sample stability without in-sample stability. That would mean that despite the fact that the different scenario trees imply rather different solutions and optimal function values, the true, out-of-sample, objective function values are (almost) the same. If that is the case, it would be possible to proceed with the model, but in a very careful manner. A solution corresponding to an arbitrary tree would then be as good a solution as that based on any other tree, but the corresponding optimal function value would not be a good measure of the true value. It should be clear that many types of testing of the model would be very difficult, though possibly not impossible, in such a setting. We would, however, not in any way recommend to proceed with a model that does not demonstrate in-sample stability.

> If you do not have in-sample stability, you possibly have not properly understood your model.

4.2.6 Out-of-Sample Stability for Multi-period Trees

So far, we have only discussed the single-period case. For multi-period trees, the situation is more complicated: We cannot simply take a solution based on one tree

and evaluate it on another one, as the nodes beyond the root do not coincide. The same is true for a simulation model: The simulated scenarios will not coincide with the tree beyond the root, so we will not be able to use the found optimal values except in the root.

This, however, is exactly what happens in reality as well: Only the first-period "root node" solution gets implemented. When we need another decision later, we re-run the model with an updated tree and implement the new root decision. This could be simulated using a rolling horizon approach, see [49] for details.

If we do not have a simulator, we can at least do the following: Build two trees $\mathcal{T}_1$, $\mathcal{T}_2$ and find the corresponding solutions $\hat{x}_1$, $\hat{x}_2$. Then solve the optimization model on tree $\mathcal{T}_1$ with the root values fixed to the root values of $\hat{x}_2$ and vice versa. If the method is out of sample stable, we should get approximately the same optimal objective values.

4.2.7 Other Approaches to Stability

There is also another approach to measuring stability of stochastic programs, by estimating the error caused by using distributions that only approximate the true one. Interested readers can have a look at, for example, Dupačová and Römisch [18] and Fiedler and Römisch [25] for two-stage problems, and Heitsch et al. [34] for multistage problems.

4.3 Statistical Approaches to Solution Quality

If you have a solution, wherever it comes from, you may want to ask how good it is relative to the unattainable optimal solution. Out-of-sample calculations can estimate the optimal value associated with a solution, and if you have several candidates, you can use out-of-sample calculations to pick the best one. You may have established in- and out-of-sample stability as we already discussed, but as there is a potential problem of bias you may be worried. If stability has not been established, the question is even more pressing: Is this actually a good solution?

Please note that this section is technically more challenging than most others in this book.

4.3.1 Testing the Quality of a Solution

So far, we have been concentrating on *stability* and avoided discussions about solution quality. The reason is that without stability, quality is not well defined as it is different every time we run the scenario generation and optimization models.

The other reason is that estimating quality of a given solution $\tilde{x}$ is difficult, as it amounts to evaluating the *optimality gap* (approximation error)

$$\mathrm{err}(\tilde{x}) = f\left(\tilde{x}; \boldsymbol{\xi}\right) - \min_x f\left(x; \boldsymbol{\xi}\right) \tag{4.3}$$

Since the right expression is the reason why we need scenarios in the first place, it might seem impossible to evaluate the above expression. There is, however, a way to derive a *statistical estimate* of the error, as long as we can sample from the underlying distribution $\boldsymbol{\xi}$, and the objective function f is an expected value, i.e.,

$$f\left(x; \boldsymbol{\xi}\right) = \boldsymbol{E}^{\boldsymbol{\xi}}\left[F(x, \boldsymbol{\xi})\right] \qquad \text{and} \qquad f\left(x; \mathcal{T}\right) = \frac{1}{N}\sum_{s=1}^{N} F(x, \mathcal{T}^s)\,, \tag{4.4}$$

where N is the number of samples (scenarios) in $\mathcal{T}$ and $\mathcal{T}^s$ is the s'th scenario in $\mathcal{T}$.

This approach comes from Mak et al. [63]: We start with a sampled tree $\mathcal{T}$ with N scenarios. From (4.4) it follows that $f\left(x; \mathcal{T}\right)$ is an unbiased estimator of $f\left(x; \boldsymbol{\xi}\right)$, i.e.,

$$\boldsymbol{E}\left[f\left(x; \mathcal{T}\right)\right] = \boldsymbol{E}\Big[\frac{1}{N}\sum_{s=1}^{N} F(x, \mathcal{T}^s)\Big] = f\left(x; \boldsymbol{\xi}\right). \tag{4.5}$$

In addition, one can show that

$$\boldsymbol{E}\left[\min_x f(x; \mathcal{T})\right] \leq \min_x f(x; \xi)\,, \tag{4.6}$$

that is, the scenario-based solutions are, on average, optimistic. This is because the scenario-based problem hedges over a restricted set of outcomes rather than the true distribution $\boldsymbol{\xi}$ and so under-estimates the true optimal expected cost.

Combining the last two equations with (4.3) gives the following:

$$\begin{aligned}
\mathrm{err}(\tilde{x}) &= f\left(\tilde{x}; \boldsymbol{\xi}\right) - \min_x f(x; \boldsymbol{\xi}) \\
&\leq \boldsymbol{E}\left[f\left(\tilde{x}; \mathcal{T}\right)\right] - \boldsymbol{E}\left[\min_x f(x; \mathcal{T})\right] \\
&= \boldsymbol{E}\left[f\left(\tilde{x}; \mathcal{T}\right) - \min_x f(x; \mathcal{T})\right],
\end{aligned}$$

which can also be written as

$$\mathrm{err}(\tilde{x}) \lessapprox f\left(\tilde{x}; \mathcal{T}\right) - \min_x f(x; \mathcal{T})\,. \tag{4.7}$$

This means that we have found a *stochastic upper bound* for the approximation error. Of course, having just one estimate of the sample error is not enough; what we need is to repeat the whole procedure for M such trees $\mathcal{T}_i$, which would give us a better estimate

$$\operatorname{err}(\tilde{x}) \lessapprox \frac{1}{M} \sum_{m=1}^{M} \left[f\left(\tilde{x}; \mathcal{T}_m\right) - \min_x f(x; \mathcal{T}_m) \right], \tag{4.8}$$

where we can replace "$\lessapprox$" by "$\leq$" if we let $M \to \infty$.

In addition to the point estimate, it is also possible to use the M values to derive a confidence interval for $\operatorname{err}(\tilde{x})$, which could also be improved by various variance reduction techniques—we refer interested readers to [63]. There are also variants of the estimate that require only two or even only one replication, see Bayraksan and Morton [5].

To get a better understanding of the quality of the estimator and how it can be improved, we can rewrite (4.8) as

$$\begin{aligned} \operatorname{err}(\tilde{x}) \lessapprox \frac{1}{M} \sum_{m=1}^{M} & f\left(\tilde{x}; \mathcal{T}_m\right) - f\left(\tilde{x}; \boldsymbol{\xi}\right) \\ & + f\left(\tilde{x}; \boldsymbol{\xi}\right) - \min_x f(x; \boldsymbol{\xi}) \\ & + \min_x f(x; \boldsymbol{\xi}) - \frac{1}{M} \sum_{m=1}^{M} \min_x f(x; \mathcal{T}_m) \end{aligned} \tag{4.9}$$

Since the second element in the right hand side is equal to $\operatorname{err}(\tilde{x})$, this gives the error of the estimator (4.8) as

$$\begin{aligned} \text{est. error} = \frac{1}{M} \sum_{m=1}^{M} & f\left(\tilde{x}; \mathcal{T}_m\right) - f\left(\tilde{x}; \boldsymbol{\xi}\right) \\ & + \min_x f(x; \boldsymbol{\xi}) - \frac{1}{M} \sum_{m=1}^{M} \min_x f(x; \mathcal{T}_m) \end{aligned} \tag{4.10}$$

The first element has zero expected value from (4.5), with variance decreasing with increasing M and the number of samples in $\mathcal{T}_m$. The second element is the *bias* of the estimator—its expected value is positive because of (4.6) and can be decreased only by improving the solutions of "$\min_x f(x; \mathcal{T}_m)$," i.e., by increasing the size of $\mathcal{T}_m$.

Another important observation is that the first element of the gap estimator (4.8) is equivalent to evaluating the solution $\tilde{x}$ using $\mathcal{T}_{N'}$ with $N' = M \times N$ samples. From (4.9), we see that this is then used as an estimate of the true objective value $f\left(\tilde{x}; \boldsymbol{\xi}\right)$; it follows that we could use any tree with $N' \gg N$ samples, i.e., use

$$\mathrm{err}(\tilde{x}) \lessapprox f\left(\tilde{x}; \mathcal{T}_{N'}\right) - \frac{1}{M}\sum_{m=1}^{M} \min_{x} f(x; \mathcal{T}_m) \tag{4.11}$$

instead. There is, however, a small catch here: Using the same samples in both elements of (4.8) results in *variance reduction*, so we would probably need to use $N' \gg M \times N$ to get results with the same variance as in (4.8).

4.3.2 Solution Techniques Based on the Optimality Gap Estimators

So far, we have used the gap estimators only to evaluate the quality of a given solution $\tilde{x}$. However, since they allow us to estimate not only how good a given solution is, but also how far from the optimal solution it is, it should not come as any surprise that these estimates form the basis for several methods for *solving* stochastic programming problems.

The most well-known of these is the so-called *sample average approximation* (SAA) method from Kleywegt et al. [57], and it works as follows: We sample M trees $\mathcal{T}_m$ with N scenarios and for each of them estimate the optimality gap (4.8) or (4.11) with some chosen N'. If both are sufficiently small, we pick the best one as our solution; otherwise, we increase M, N, or N' and repeat the whole procedure. There are, of course, many important details such as the choice of M, N, and N', and the exact meaning of "sufficiently small," for which we refer interested readers to the original paper.

Another such method can be found in Bayraksan and Morton [6]. This method builds on the assumption that we can get a sequence of feasible solutions x_k that converges to the optimal solution of (4.2)—such as solutions of the scenario approximations (4.1) with increasing sample sizes. The method itself is then similar to the SAA method with M equal to one: For a given k, we test whether the solution x_k is "good enough" using one tree with N_k scenarios. If yes, we stop; if not, we generate the next solution x_{k+1} and repeat the procedure. Note that since this method estimates gaps using only one tree at each step, it is based on gap estimators from [5] instead of (4.8).

It may appear that these methods always work. But that is not the case and should in fact not surprise you: Your problem may be so large or complicated that there is no way with today's technology we can find good solutions. The methods assume that there is a reasonable chance that a sampled tree leads to a good solution. But for a really large problem that probability might be practically zero (as you really need scenario trees vastly larger than what you can handle), so you just keep wandering around in darkness.

Sample-based methods using statistical approaches to check solution quality may not work if the problem is too large.

An interesting observation about the above method is that it is not important where the solutions come from—we only need to guarantee that we have a sequence converging to the optimal solution. So if we have a scenario generation method that is better than sampling that converges to the true distribution, it is probably a good idea to use it instead. But what if we have a scenario generation method that is, according to the tests presented in this chapter, better than sampling for the range of tree sizes we are able to solve, but we do not know if it converges to the true distribution (or even know it does not)? Can we still use it to generate the solutions, instead of using sampling? We are not aware of any studies on this subject, but we dare guess that the answer is "yes"—scenario tree quality for the tree sizes we actually use should be more important than the limit behavior we never observe.

4.3.3 Relation to the Stability Tests

At first sight, this statistical approach does not seem to be in any way related to the stability we talked about earlier in this chapter. This, however, is not correct: There is a very close relationship between these two, as we will show below.

The statistical tests help us to evaluate the *bias* from Sect. 4.2. There, we first require a given scenario generation model to be *stable* because otherwise we cannot say what a "solution" or "optimal objective function value" is, as they are different on every run of the model. Then we explain that stability is not sufficient to guarantee good solutions, since a wrongly selected scenario generation method can still lead to sub-optimal solutions of the optimization problem—in which case we called the scenario generation method "biased." However, back in Sect. 4.2.3, we did not have any tools to estimate the bias. The statistical tests from Sect. 4.3 allow us to estimate at least an upper bound on the optimality gap (bias) and in this way complement nicely the other tests from Sect. 4.2.

The statistical tests also lend support to the intuitive idea that having a big difference between the in- and out-of-sample objective values $f\left(\hat{x};\mathcal{T}\right)$ and $f\left(\hat{x};\boldsymbol{\xi}\right)$, for a solution $\hat{x} = \min_x f(x;\mathcal{T})$, is not a good sign: From (4.7), we see that the difference is a stochastic upper bound on the optimality gap,

$$f\left(\hat{x};\boldsymbol{\xi}\right) - f\left(\hat{x};\mathcal{T}\right) = f\left(\hat{x};\boldsymbol{\xi}\right) - \min_x f(x;\mathcal{T}) \gtrapprox \mathrm{err}(\hat{x})\,.$$

We should, of course, keep in mind that since it is a upper bound, a big difference does not necessarily mean big optimality gap; as we can see from (4.9) with $M = 1$, the difference can also be caused by the bias of the gap estimator.

In addition, we can see what the stability concepts tell us about the gap estimator itself. In this context, the *in-sample stability* from Sect. 4.2.1 refers to the variance of "$\min_x f(x; \mathcal{T}_m)$" from (4.8); if it is big (i.e., if we do not have stability), then the whole estimator (4.8) will have a big variance and hence a confidence interval that is too wide for the estimate to be useful.

Similarly, the *out-of-sample stability* from Sect. 4.2.2 refers to the variance of $f\left(\hat{x}_m; \boldsymbol{\xi}\right)$, approximated by $f\left(\hat{x}_m; \mathcal{T}_{N'}\right)$—which is the first element in the optimality gap estimator (4.11). Again, it follows that out-of-sample instability implies big variance of the gap estimator, with the same results as in the in-sample case. Finally, the *bias* from Sect. 4.2.3 does not exist in this context, since the estimators use sampling, which is unbiased—see (4.5).

4.4 Property-Matching Methods

We mentioned earlier that we should measure the quality of a scenario tree using the objective function of the stochastic program as metric. That is, if two discretizations give the same expected cost (or profit), they are equally good, and the optimal discretization is one that trades off problem size (the number of scenarios) and quality of solution as well as possible. So for a given number of scenarios, we want as good an approximation of expected costs as possible, or, for a given quality of the expected cost, we want as few scenarios as possible. This is reflected in our quality tests, and in the previous section, they all use the objective function to measure quality.

In Sect. 4.2 we showed how to test the quality of a scenario generation method by studying stability. The idea was to establish a situation where you could safely run your scenario generation method only once and then solve the stochastic programming problem, being confident that the solution you got made sense. This scenario generation method could be sampling, but we indicated many times that for a reasonably large problem sampling is not likely to lead to stability. Rather, for sampling, it is more reasonable that you need results from Sect. 4.3 to check the quality of the solutions you obtain.

If you have your own scenario generation method, you are advised to use the stability tests first to see if you can avoid the heavy work of testing the quality of solutions you obtain. If you cannot establish stability, then testing quality is your only option if you at all care about the meaning of what you have obtained.

In this section we shall present an approach for generating scenarios that is an alternative to pure sampling and that tries to take into account that the quality of a scenario tree should be measured by the stochastic program you are solving, not by purely comparing distributions. Although the tests in Sects. 4.2 and 4.3 use the optimization problem to measure stability and quality, if you used sampling to generate the trees, you did not use a scenario generation procedure that tried to fit the test; when you sample, you indirectly care about all aspects of a distribution, also those that do not matter to the optimization problem and hence the tests. This

is no problem if you end up with positive results. But it is rather likely that this lack of match between how you generate and how you measure scenarios is a cause for trouble.

But is it better to match moments (and other statistical properties) than to sample and in fact try to match *all* properties? The answer is: Only if you end up regarding those properties that matter. How can we know that apart from special cases like the Markowitz model we illustrated in Fig. 4.1? Rarely can we do it based on such direct analysis. But we can use our brains a bit:

- The means and variances are almost always important.
- If we use a risk measure that is not symmetric (contrary to the variance that is symmetric), the third moments are very likely important.
- If we care about the probability of really bad outcomes, we may have to look at the fourth moments that describe the thickness of the tails.
- We need to ask if the distributions are elliptic (like the normal) so that correlations are likely to describe co-variation well. If the answer is no, we need to consider whether or not that matters.

In what follows we shall first discuss how you might think about formulation a model that produces a scenario tree matching a variety of possible properties (not just moments). Numerically that approach is not very good in light of stability, but we show it as it illustrates the idea behind property (moment) matching. Then we pass to an approach that looks at only four marginal moments plus correlations. That method is efficient and simple to use. It is our experience that it very often leads to stability—but that must of course be tested.

If you do not get stability with this approach, there are two possibilities: Make the tree larger or revise your choice of properties. There are no automatic ways of doing the latter, and you need to use problem understanding and simply try. There is of course a reasonable chance that also for this kind of approach, just as for pure sampling, you need so many scenarios to get anything meaningful, that the problem cannot be solved. Do not be surprised or discouraged by this. Some problems are simply so large that it is not possible with today's technology to solve meaningful stochastic versions of them. In this case you will either have to develop stronger algorithms for solving stochastic programs, probably specialized for your problem (which certainly is a feasible path in many cases), or you will have to revise and simplify your model.

4.4.1 Regression Models

One way to set up a scenario tree is to define properties you want to be present in the tree, and then, for example by non-linear programming, create a tree with those properties. In such a problem, the variables are the outcomes of the random variables and possibly also the probabilities. The ideas here are taken from [42].

Let $\boldsymbol{\xi}$ be the discrete random vector we are to generate. A single random variable is then denoted $\boldsymbol{\xi}_i$, for $i \in \mathcal{I}$. A scenario is a vector ξ with individual elements ξ_i for $i \in \mathcal{I}$. If we need to index a scenario, it will show up as ξ^s with elements ξ_i^s for $s \in \mathcal{S}$. Assume you want a specific mean μ in your scenario tree. Hence, you want

$$\sum_{s \in \mathcal{S}} p^s \xi^s = \mu.$$

You may think of this as either a constraint in a non-linear optimization problem (with ξ and p as variables), or you may put the expression into the objective function with a penalty for violation.

Variances could be controlled by

$$\sum_{s \in \mathcal{S}} p^s \, (\xi_i^s)^2 - \big(\sum_s p^s \, \xi_i^s\big)^2 = \sigma_i^2 \, .$$

In the same way, you may proceed with any expression that can be written as a function of ξ and p. For example, if you need to ensure that in at least one scenario, $\boldsymbol{\xi}_i = \underline{\xi}_i$, the relevant expression is

$$\xi_i^1 = \underline{\xi}_i \, .$$

As the indexing of the scenarios is arbitrary, we have simply chosen to let the extreme value of $\boldsymbol{\xi}_i$ be in scenario 1. Any choice would be OK. As a final example, if you wish the correlation between $\boldsymbol{\xi}_i$ and $\boldsymbol{\xi}_j$ to be r_{ij}, you need

$$\sum_{s \in \mathcal{S}} p^s \, \xi_i^s \xi_j^s - \mu_i \mu_j = \sigma_i \sigma_j \, r_{ij} \, ,$$

as long as also means and variances are controlled.

If the probabilities are variables, you also need to add $p^s \geq 0$ and $\sum p^s = 1$. Numerics may dictate that it is simpler to let the probabilities be fixed in the scenarios, typically at $p^s = 1/|\mathcal{S}|$. The price to be paid for this simplification is that you may need more scenarios to describe what you want.

Whatever choices you have made, you now need to find a set of p^s and ξ^s that is feasible (with respect to whichever expressions you have chosen to add as constraints) and with zero as the optimal objective value. If that is achieved, you have a scenario tree with exactly the required properties.

But before embarking on a regression-based approach, see if you can utilize the special situation described in Sect. 4.4.2. The reason for that is two-fold. First, it is much simpler numerically. Second, the approach from Sect. 4.4.2 produces better results. The reasons for this are discussed under "A different interpretation of the model" on page 100. However, regression models form the basis for all these ideas, and there might be cases where that is your only choice, as you need to represent distributional properties for which there are no simple approaches.

4.4.2 The Transformation Model

This is an iterative procedure that produces trees with specified first four marginal moments and correlations outlined in [43]. It is based on transformations and is very quick. Although there has been some work on extending the method beyond the four moments and correlations, there is clearly a limit to how far it can be extended.

The procedure consists of three parts: An initialization, and two transformations that are repeated in an iterative loop.

4.4.2.1 Initialization

The procedure needs a starting distribution. Although, in principle, you can start many different ways, we usually start with independent standard normal random variables $\boldsymbol{\xi}_i$.[1] While it would be easy to obtain the values for each margin by sampling, we prefer to use a fixed discretization to increase the "smoothness" of the distribution. For discretization with $|\mathcal{S}|$ scenarios, we would typically divide the support into $|\mathcal{S}|$ intervals, each having the same probability, namely $1/|\mathcal{S}|$. The outcome associated with an interval is the conditional expected value. Then, create scenarios $\xi^s = (\xi_1^s, \xi_2^s, \ldots, \xi_{|\mathcal{I}|}^s)$ by randomly combining outcomes from the different discretizations. Here ξ_j^s is one of the outcomes for random variable j. Since the combinations are random, all correlations in the tree will have zero expectations. However, since the tree is of limited size, the actual correlations will be somewhat off zero.

So, with this approach, the starting tree has approximately uncorrelated standard normal margins. Note that if you have other ways of initiating the distributions, for example, ways of better representing co-variation among the random variables, you should test them.

4.4.2.2 Correcting Correlations

The first transformation changes a tree to obtain the desired correlations. To do it, we use the standard method for generating correlated normal variates: If $\boldsymbol{x}$ is a vector of uncorrelated random variables, and R is a correlation matrix with Cholesky decomposition $R = LL^T$, then $\boldsymbol{y} = L\boldsymbol{x}$ has a correlation matrix equal to R.

If $\boldsymbol{x}$ was a vector of standard normal variables, the margins of $\boldsymbol{y}$ would again be standard normal. In the general case, however, the transformation changes the marginal distributions (and hence the moments). The only thing we can guarantee is that if all the margins of $\boldsymbol{x}$ have zero mean and variance equal to one, the same

[1] We do not recommend using the starting distribution described in [43], as it seems to have a negative effect on the quality of resulting scenarios—not to mention that it does not have to exist in the first place.

will be true for the margins of $\boldsymbol{y}$. To take advantage of this, all the random variables are standardized (set to zero mean and variance equal to one) inside the algorithm. They are transformed to their specified means and variances at the very end of the algorithm, using a simple linear transformation $\boldsymbol{y}_i \leftarrow \mu + \sigma\, \boldsymbol{y}_i$.

The transformation $\boldsymbol{y} = L\boldsymbol{x}$ works only if the margins of $\boldsymbol{x}$ are uncorrelated. In the first iteration, they may be close, in the sense that we expect all correlations to be zero. In later iterations, that is certainly not the case. Assume we want $\boldsymbol{y}$ with correlation matrix $R = LL^T$ but have $\boldsymbol{x}$ with correlation matrix $R_k = L_k L_k^T$. Then we know that $\boldsymbol{z} = L_k^{-1}\boldsymbol{x}$ is uncorrelated, and

$$\boldsymbol{y} = L\boldsymbol{z} = (LL_k^{-1})\boldsymbol{x}$$

has the required correlation matrix R. Hence, it is the matrix LL_k^{-1} that is used in the computations. The index k here plays the role of an iteration counter.

Note that the Cholesky components L and L_k have a "1" in the top-left corner, so the same must be true for the lower-triangular matrices L_k^{-1} and LL_k^{-1}. This means that the first margin is never changed by the transformation.

4.4.2.3 Correcting Moments

This is the most computing intensive part of the algorithm. A cubic transformation of the outcomes in the tree changes the tree into another tree with the required first four marginal moments. The setup is that we, for each random variable i, need to find four parameters a_i, b_i, c_i, d_i such that

$$\boldsymbol{y}_i = a_i + b_i\boldsymbol{x}_i + c_i\boldsymbol{x}_i^2 + d_i\boldsymbol{x}_i^3$$

has the specified first four moments. The parameters are found by solving a system of four implicit equations in four unknowns. For details on the equations, we refer to [43].

Since we use a non-linear transformation, it will change the correlations, leaving us with margins with the correct moments and a slightly off correlation matrix—so we use the correlation-correcting transformation again. This is repeated a few times until both moments and correlations are in place.

4.4.2.4 Illustrative Example

As an example of the algorithm, we generate 200 scenarios for two random variables with correlation 0.5. The first four moments of the first one are {0,−1,0.75,4}, the

other {0,1,0.75,4}.[2] Since the first variable does not change by the correction of correlations (see above), it remains unchanged once its moments are corrected in the first iteration. We thus present only a histogram of the second variable, together with a scatter-plot to show the two-dimensional structure.

The results of the algorithm are in Fig. 4.2. The first line shows the distribution after the first correction of the correlation, so the margins are still normal. The second line presents the situation at the end of the first iteration, after the correction of the marginal distributions. Note that the correlation is slightly distorted. The last two lines show the corresponding results of the second iteration. We can see that correcting the correlation slightly distorted the margins, but the following correction of the moments changed the correlation so slightly that it remained correct up to the second decimal place.

With only 200 scenarios, it is impossible to present the shape of the density, so we have done the same test with 10,000 scenarios. The algorithm was again stopped after the second iteration, and the density was estimated using a kernel estimator. The result is in Fig. 4.3.

4.4.2.5 Does It Always Work?

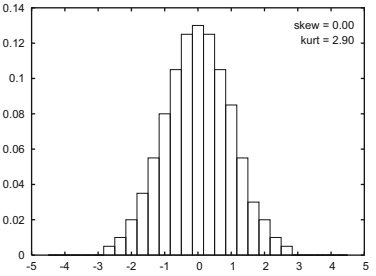
There is no proof that the above iterative procedure works in all cases. However, there is no reason to be very concerned. It is very easy to check after the procedure has stopped what are the statistical properties of the resulting tree. Hence, there is no danger that we end up using a tree with incorrect properties.

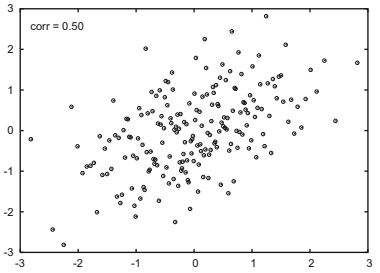
But is it possible that the transformations will yield incorrect results? So far, we have never observed lack of convergence, except in two situations. The first is that the properties used to define the tree are internally inconsistent, that is, that there exists no tree like the one asked for. It is possible to define values for the marginal moments that do not correspond to anything real. This will of course never happen if the numbers are calculated from a single actual data set. But if they are simply defined, that could happen. The second case that might result in an incorrect tree is that the tree we ask for is too small. Clearly, the more properties you ask for, then more variables (i.e., scenarios) you need.

Hence, if you observe lack of convergence, the obvious first attempt around it would be to make the tree larger, and try again. If that does not work, it is worth checking if the required properties make sense at all. Although never observed, there is of course (as long as we have no proof) a chance that lack of convergence may show up even with a well-defined distribution and large enough tree.

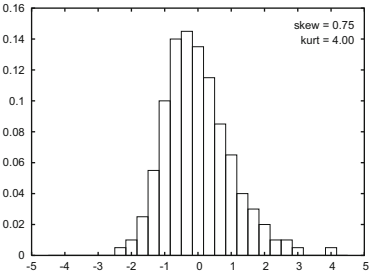
In the end, the only way to find out how the method works for *your* problem is to test it yourself. It is freely available from https://work.michalkaut.net/#HoylandEA03, including the source code.

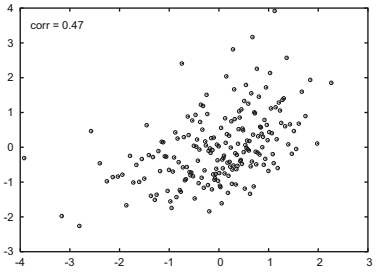
[2] For moments of order $k > 2$, we use $\mathrm{mom}_k(\mathbf{X}) = \boldsymbol{E}\left[(\mathbf{X} - \mu)^k\right]/\sigma^k$.



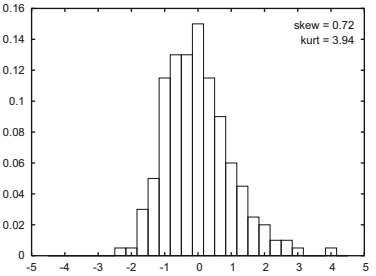


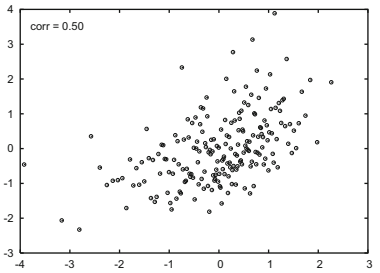
Iteration 1, after correction of the correlation.



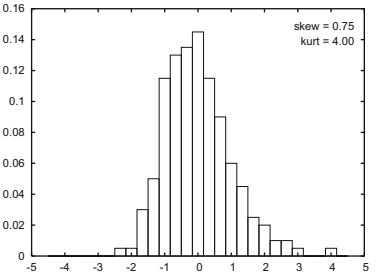


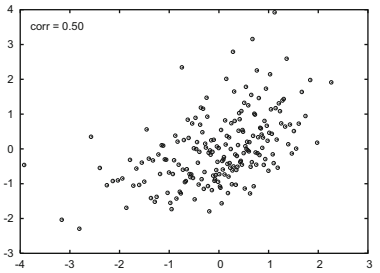
Iteration 1, after correction of the moments.





Iteration 2, after correction of the correlation.





Iteration 2, after correction of the moments.

Fig. 4.2 First two iterations in a case of two correlated variables

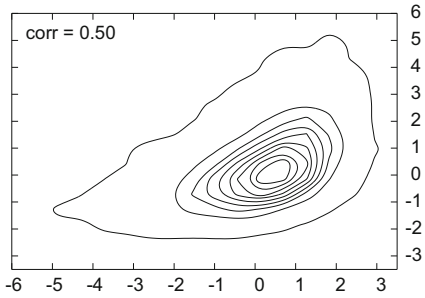


Fig. 4.3 Iteration 2, after correction of the moments—10,000 scenarios

4.4.2.6 A Different Interpretation of the Model

The interpretation given so far of the transformation model is simply that it is an efficient way of solving the regression model for a specific case, namely four marginal moments and correlations. However, that may not be the most useful way of interpreting the method. Several researchers have claimed that matching four marginal moments and correlations does not work (meaning does not give useful results). Why this apparent disagreement?

The reason is that the transformation model does much more than match four marginal moments and correlations. First of all, it allows (and encourages) to generate more scenarios than what is needed to just match the moments and correlations. Relative to the original regression model (which does not use extra degrees of freedom in any useful way), the transformation model uses this freedom to achieve properties we have not asked for.

For once, starting with a "nice" discretization ensures that many unspecified properties of the distribution have "reasonable" values. The transformations involved in the algorithm preserve at least some of the properties and, more importantly, leave these properties almost the same from iteration to iteration of the scenario generation procedure.

The resulting marginal distributions are "smooth"—as a visual statement of how they look. Hence, the resulting scenario tree can also be seen as an approximation of the whole distribution, with some properties guaranteed and some at least having "reasonable" values. And it is this total package, rather than just the limited set of properties, that produces stable models.

You may be disturbed by the fact that we do not quite know what is going on. It would certainly be better if we knew, but one of the purposes of Sect. 4.2 is to test (rather than simply discuss) if a certain procedure produces stable results. And it turns out that for many models, the transformation model produces stable results, while other methods that match four marginal moments and correlations do not.

4.4.2.7 Extensions of the Model

While the method works satisfactorily for many models, there are situations where four moments and correlations simply do not provide sufficient control over the distribution. As an example, the model might be sensitive to the shape of the multivariate distribution, in which case correlations would not provide enough control. Recently, there has been some development to cope with both possibilities.

In the case when we need more than just correlations to control the co-variation, [50] discusses controlling the shape of the multivariate distribution using the notion of copulas. Unlike the previous case, which basically generalizes the transformation method, here the approach is rather different. In addition, the proposed methods do not allow for an exact control of correlations, though this can be solved by using the transformation method as a post-process.

4.4.2.8 Alternative Approach Using Copula

While the method works satisfactorily for many models, there are situations where four moments and correlations simply do not provide sufficient control over the distribution. As an example, the model might be sensitive to the "shape" of the multivariate distribution, in which case correlations would not provide enough control.

For these cases, [50] discuss an approach where the shape of the multivariate distribution is controlled using *copula*, instead of a correlation matrix. This is combined with pre-generated discretizations of the marginal distributions: The copula is basically a function that tells us how to join the marginal distributions together into a multivariate distribution. Unfortunately, discretizing multivariate copulas is difficult problem, and we are not aware of any method that could do it in a general case. Instead, [48] presents a heuristic that discretizes a set of 2D copulas, instead of the full multivariate copula. For many cases, this method produces better scenarios than the moment-based heuristic, at the expense of a longer generation time.

In addition, the copula-based methods by definition do not allow for an exact control of correlations, though this could be addressed by using the transformation method as a post-process.

4.4.3 Independent and Uncorrelated Random Variables

In some cases the random variables are genuinely independent. However, more commonly we get independent or uncorrelated random variables when we apply such algorithms as principle component analysis to the underlying randomness. In some respects independence is easier to handle, but most of the time it is not.

The case where independence is easy is as follows. Assume you have n random variables, each taking on k possible values. If we make scenarios by combining all-against-all, arriving at k^n scenarios, it is very easy to calculate scenario probabilities in the independent case.

But we can rarely allow ourselves the luxury of having k^n scenarios. Hence, we must resort to ideas like those presented in this chapter. And then independence is a property we *enforce* on the scenarios (in fact, unless we use the copula-based method, we are only able to enforce zero correlations). Since the method, as explained in this chapter, starts out with almost zero correlations, it will converge quickly in that case. But otherwise, independence (or zero correlations) is not a major issue, any correlation structure can be enforced by this method.

4.4.4 Other Construction Approaches

In his famous paper [68], Pflug showed that the approximation error can be bounded by the distance between the discretization and the original distribution, expressed using the Wasserstein metric (transportation distance), and proposed to generate scenarios by constructing discretizations that minimize this distance. This gave rise to a series of papers about generating or improving scenario trees by minimizing this—or some related—metric; the most well-known are the papers about *scenario reduction* methods, see Dupačová et al. [17], Heitsch and Römisch [32, 35] for two-stage and Heitsch and Römisch [33] for multistage models. These methods give much better results than pure sampling, for a given number of scenarios.

Note however that these methods focus purely on some distance between the two distributions; the optimization problem appears in the definitions of the approximation error but disappears in the bounding process. This means that these methods will always try to replicate the whole distribution, instead of focusing only on the properties important for the given optimization model (like mean and (co)variances for the Markowitz model)—and that does not come for free in terms of the number of scenarios. On the other hand, removing the optimization model from the scenario generation process can be seen an advantage, as it means that scenario generation can become a specialty in its own right, not requiring optimization knowledge. Many prefer these approaches for that reason.

In the end, the decision is the same as for so many other methods: We should test if they work for our problem, using either the stability or statistical tests, and if they do then all is fine.

4.5 Problem-Driven Scenario Generation

In Sect. 4.1 we stated that the quality of a scenario tree depends on the problem one is trying to solve. The question that naturally follows from this is: How do

we generate scenarios in a way that is appropriate to the problem? *Problem-driven* scenario generation methods are ones that exploit the underlying structure of a problem to provide a more concise representation of the uncertainty.

Most scenario generation methods we have encountered so far have been *distribution-driven*. That is, they aim to construct a scenario tree that has certain statistical properties or that approximates some other distribution. Although property-matching methods in Sect. 4.4 should be problem-driven (they require analysis of problem to select appropriate properties), they are often in reality used as a convenient way of modeling uncertainty and do not take the problem into account.

In this section we first present in detail a problem-driven approach and draw some general observations from this which apply to other problem-driven approaches as well. We then provide an overview of other problem-driven approaches in the literature. Throughout this subsection, we will assume in our stochastic program that we are minimizing the expectation or some other risk measure of some general *loss function*.

4.5.1 Scenario Generation for Stochastic Programs with Tail Risk

This methodology applies to problems that use tail risk measures, such as CVaR (see Sect. 3.5), that only depend on the upper tail of a distribution, and was originally developed in [22, 23]. For simplicity we present its application to the portfolio selection problem that was also discussed in Chap. 3. Recall that in this problem we have a collection of assets indexed by $i = 1, \ldots, n$ and must decide on x_i, how much to invest in each asset. The return of each asset i is a random variable and denoted by $\boldsymbol{\xi}_i$. The total return corresponding to a portfolio $x = (x_1, \ldots, x_n)$ and returns $\xi = (\xi_1, \ldots, \xi_n)$ is then given by $\sum_{i=1}^{n} x_i \xi_i$. However, since we are interested in the risk of large losses, we work with the negation of this value. In other words, the loss function in this problem is

$$-\sum_{i=1}^{n} x_i \boldsymbol{\xi}_i$$

The aim of this stochastic program is to minimize the CVaR, or some other tail risk function, of this loss function, subject to some deterministic constraints like no short-selling and investing all capital. In other words, our objective is to choose a portfolio that mitigates against the outcomes with worst-case returns.

Let us consider now how the CVaR is calculated for a given portfolio. In Fig. 4.4 is plotted equally probable return scenarios for two hypothetical assets. Consider the portfolio $x = (0.5, 0.5)$, that is, the portfolio where we invest equally in both assets. This portfolio induces a distribution of losses $-0.5\boldsymbol{\xi}_1 - 0.5\boldsymbol{\xi}_2$. The scenarios that are used for the calculation of the β-CVaR are those with loss below the β-quantile

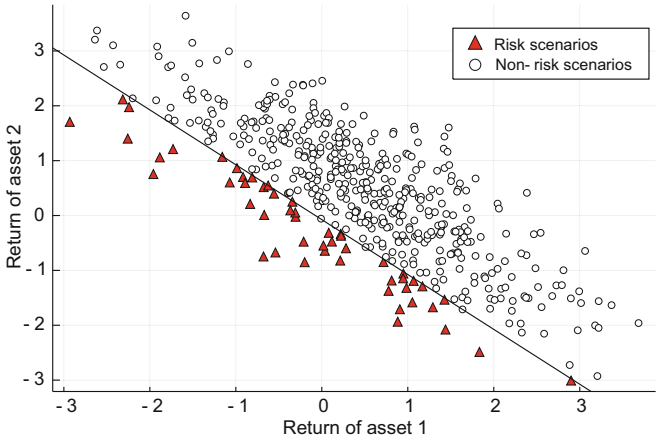


Fig. 4.4 Loss scenarios of two hypothetical assets. Scenarios below the line have a loss below the 90% quantile for the portfolio $x = (0.5, 0.5)$

of loss, which correspond to scenarios where the assets have large negative returns. More precisely, denoting the β-quantile of the loss function by q_β, the scenarios (ξ_1^s, ξ_2^s) that are used in the calculation of the β-CVaR are the ones that satisfy the following inequality:

$$-0.5\xi_1^s - 0.5\xi_2^s \geq q_\beta.$$

We have also plotted the line $-0.5\xi_1 - 0.5\xi_2 = q_\beta$ in Fig. 4.4. We call the scenarios below this line, *risk scenarios* as these and the ones with the worst performance for this portfolio.

This exercise can be repeated for all feasible portfolios in our problem. In this example, we will assume the feasible portfolios to consist of all non-negative portfolios.[3]

In Fig. 4.5 we have plotted quantile lines for all feasible portfolios. In this way, we have identified all the scenarios that are important for our problem, that is, all scenarios that have poor performances for some portfolio. These are precisely the scenarios necessary to calculate the β-CVaR for all portfolios. The remaining scenarios are not needed to calculate the CVaR for any portfolio and thus are not needed for the problem of optimizing the CVaR over all feasible portfolios. We can therefore aggregate the non-risk scenarios into a single scenario (to preserve the quantiles) without affecting the values of the CVaR. This is shown in Fig. 4.6. This reduced scenario set leads to a smaller and easier to solve optimization problem.

We call the region containing the risk scenarios the *risk region*.

[3]That is, portfolios in which we cannot short assets.

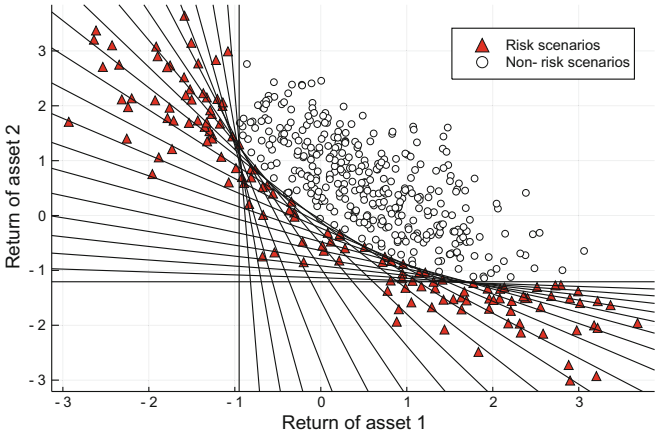


Fig. 4.5 Return scenarios for two hypothetical assets. Loss quantile lines have been plotted for all non-negative portfolios

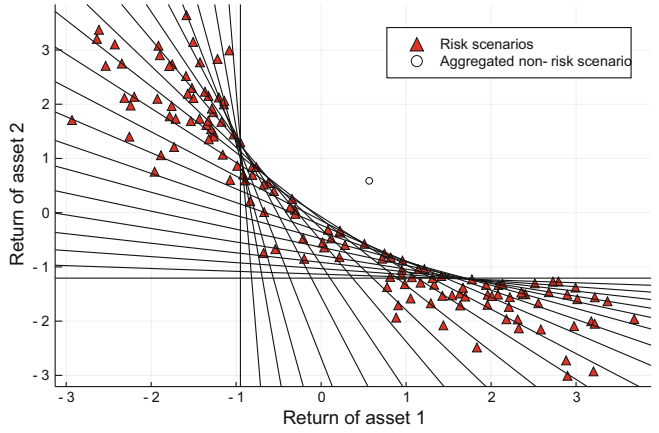


Fig. 4.6 Return scenarios for two hypothetical assets. All non-risk scenarios have been aggregated into a single point

It was shown in [23] that, under mild conditions, all the mass outside of this set (the *non-risk region*) can be aggregated into a single scenario without affecting the value of the tail risk measure. Moreover, by aggregating the mass in the non-risk region into the conditional expectation, we also preserve the overall expected loss. This result is a concrete example of the idea expressed earlier that some parts of a distribution will be more important than others.

As well as being used to reduce a given set of scenarios, this idea can also be used to generate new scenario sets. Assuming the random vector of asset returns follows a distribution from which we can sample, we can draw samples, keeping those in the

risk region, and aggregating those which are not, until we have the specified number of scenarios in the risk region.

The difficulty of applying this methodology to any problem with a tail risk measure is finding a characterization of the risk region for which we can easily test membership. In general, the risk region depends on the loss function, distribution, and constraints of the problem, and it is not always possible to derive a convenient analytical expression for it. However, for the portfolio selection problem with a normally distributed loss, $\boldsymbol{\xi} \sim \mathcal{N}(\mu, \Sigma)$, and a set of feasible portfolios $\mathcal{X} \subset \mathbb{R}^n$, we are able to give the following characterization:

$$\mathcal{R}_{\mathcal{X}} = P^T \left(\{ \xi \in \mathbb{R}^n : \| p_{K'}(\xi - \tilde{\mu}) \| \geq \Phi^{-1}(\beta) \} \right)$$

where P is such that $\Sigma = PP^T$, $\tilde{\mu} = (P^T)^{-1}\mu$, $K' = -PK$, $K = \text{conic}(\mathcal{X})$, and Φ^{-1} denotes the inverse cumulative distribution function of a standard normal distribution. The function $p_{K'}$ projects points onto the cone K'. The proof of this result is somewhat complicated, so we refer the interested reader to [23] for more details. This is a complicated expression, but the important point here is that testing membership of the risk region involves linear transformations, and projecting onto a cone. Although these are fairly small calculations compared with the problem at hand, it requires significantly more computation than simpler scenario generation methods such as sampling. This is a common feature of problem-driven scenario generation methods.

Problem-driven scenario generation methods tend to be more computationally expensive than distribution-driven methods.

Fundamentally, to use this scenario generation approach, a convenient characterization of the risk region or approximation of this needs to be derived for the problem at hand. Although [22, 23] give some other examples of how risk regions may be used or approximated for other problems, this will not always be possible. This issue of applicability to other problems is something which afflicts many problem-driven scenario generation methods, and it underlines the difficulty in sometimes applying these methods:

A problem-driven scenario generation methodology developed for a particular class of problems may need to be adapted for specific instance of a problem. In some cases, it may be impossible to apply the method to a particular instance.

Another important feature of the risk region approach is that the smaller the set of feasible portfolios, the smaller the risk region, that is,

$$\mathcal{X}' \subset \mathcal{X} \implies \mathcal{R}_{\mathcal{X}'} \subseteq \mathcal{R}_{\mathcal{X}}.$$

This follows since the more feasible portfolios we have, the more scenarios that could perform poorly with respect to one of these portfolios. The practical upshot of this is that for smaller feasible sets, a smaller part of the distribution needs to be approximated.

We illustrate this for the portfolio selection problem in Fig. 4.7 for two assets that we assume, for simplicity, to have returns that follow independent standard normal distributions. We sample an initial scenario set of 250 scenarios that is shown in Fig. 4.7a and then reduce this set for three different sets of constraints for $\beta = 0.95$.

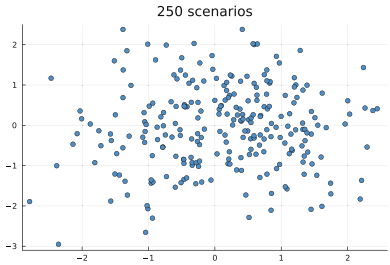
In the first case, shown in Fig. 4.7a, the only constraint we have is that we have to invest in something. In particular we allow negative investments, that is the short-selling of assets. In the case where we short both assets, positive returns in the two assets lead to losses, and so scenarios where both assets have very high returns are in the risk region. Only the scenarios where the assets have moderate losses or returns can never lead to large losses in the portfolio, and so these are the scenarios that can be aggregated in this case. The amount of scenarios we reduce here with respect to the original scenario set is about 72%.

In the second case, shown in Fig. 4.7b, we include non-negativity constraints, that is, we forbid short-selling. In this case our portfolio results in a loss only if we have negative losses in the assets we invest in. As a consequence, only scenarios with large negative returns in either of the assets can lead to extreme losses, and so the scenarios in the risk region are now confined to the negative quadrants. The amount of scenarios we can reduce in this case therefore increases to 86%.

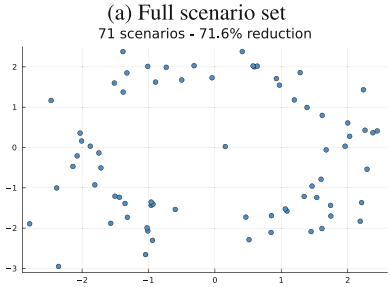
In the final case, shown in Fig. 4.7c, we include non-negativity constraints, and constraints that preclude investing too much in either asset. Scenarios in which we have a large negative return in one asset but a moderate loss or positive return in the other are now no longer in the risk region as our constraints mean we are never completely exposed to the losses in a single asset. Reduction in this case increases a small amount to 90% of the original scenario set.

The advantage of having tighter first-stage constraints for the purposes of scenario generation is something that in principle extends to general stochastic programming problems. The tighter the first-stage constraints of a problem, the fewer decisions for which we must approximate the distribution of the recourse function, and so the fewer scenarios that are needed to approximate this well. To summarize:

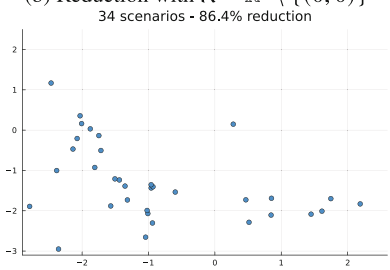
> The more constrained a problem, the greater the potential for problem-based scenario generation methods.



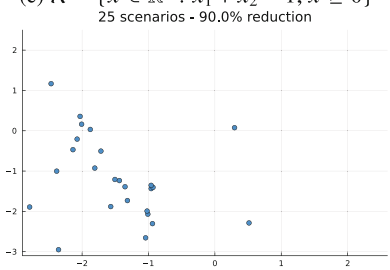
(a) Full scenario set



(b) Reduction with $\mathcal{X} = \mathbb{R}^2 \setminus \{(0,0)\}$



(c) $\mathcal{X} = \{x \in \mathbb{R}^2 : x_1 + x_2 = 1, x \geq 0\}$



(d) $\mathcal{X} = \{x \in \mathbb{R}^2 : x_1 + x_2 = 1,\ \frac{1}{2}x_1 \leq x_2 \leq 2x_1,\ x \geq 0\}$

Fig. 4.7 Scenario generation for portfolio selection problem with a tail risk measure under tightening constraints. (**a**) Full scenario set. (**b**) Reduction with $\mathcal{X} = \mathbb{R}^2 \setminus \{(0,0)\}$. (**c**) $\mathcal{X} = \{x \in \mathbb{R}^2 : x_1 + x_2 = 1, x \geq 0\}$. (**d**) $\mathcal{X} = \{x \in \mathbb{R}^2 : x_1 + x_2 = 1,\ \frac{1}{2}x_1 \leq x_2 \leq 2x_1,\ x \geq 0\}$

However, be aware that not all problem-driven scenario generation methods will be able to exploit constraints in this way.

The approach described here is not the only approach to scenario generation for stochastic programs with tail risk. The papers [3, 71] both provide alternative

scenario reduction approaches to problems using the conditional value-at-risk tail risk measure in particular.

4.5.2 Other Problem-Driven Approaches

We described above in detail one specific example of a problem-driven approach to scenario generation, but many others exist. We review some of these alternative approaches here. We assume that scenario generation is for a stochastic program where one is minimizing the expectation of some loss function subject to deterministic constraints, that is, a problem of the form:

$$\min_{x \in \mathcal{X}} \mathbb{E}\left[F(x, \boldsymbol{\xi})\right]. \tag{4.12}$$

Importance Sampling

The oldest problem-driven approach to scenario generation is the importance sampling method developed in Dantzig and Infanger [14, 44]. In importance sampling, one samples from an alternative distribution to prioritize the construction of scenarios in parts of the distribution deemed to be most important to the problem. These samples are then reweighted in order to have an unbiased estimate of the objective, that is, the expected loss in (4.12). This is a variance reduction technique, but care must be taken in constructing an appropriate sampling distribution, as a bad choice can actually lead to higher variance estimates of the objective function.

When the objective function of the stochastic program is the expectation of some loss function as above, the optimal choice of sampler for estimating the objective function at a particular decision is one whose probability density is proportional to the product of the loss function and the original probability density. This requires the loss function to be non-negative but moreover assumes one already knows the expected value of the loss function at the given solution, which is the value we are trying to estimate in the first place, and thus unavailable. Therefore approaches to importance sampling for scenario generation try to approximate this optimal sampler.

In Dantzig and Infanger [14, 44], this is done by using additive approximation of the loss function. In the more recent paper, [66] a Markov Chain Monte Carlo approach is used to produce an approximate sample from the optimal sampler. These methods typically involve evaluating the recourse function repeatedly, and so this method is potentially quite computationally intensive to use if the second-stage problem is difficult to solve. Note also that the sampler used in an importance sampling algorithm typically depends on a first-stage decision. These techniques therefore tend to be used in solution algorithms that include sampling as part of the solution procedure, rather than to generate a single-scenario tree.[4]

[4] See the end of the introductory section of this chapter for a brief discussion of such algorithms.

Regression-Based Approaches

Regression approaches to scenario generation are approaches that for a given set of first decisions $x_1, \ldots, x_K$ called a *solution pool* explicitly aim to preserve the expected loss function as far as possible with respect to some "true" random vector $\boldsymbol{\xi}$. That is, they aim to select scenarios $\xi_1, \ldots, \xi_N$ and corresponding probabilities $p_1, \ldots, p_N$ such that:

$$\sum_{s=1}^{N} p_s F(x_k, \xi_s) \approx \mathbb{E}\left[F(x_k, \boldsymbol{\xi})\right] \qquad \text{for } k = 1, \ldots, K.$$

We refer to values on the left-hand here as in-sample values and the right-hand side as out-of-sample values. The rationale for such an approach is that preserving the objective function for some solutions should also preserve it for other solutions near these. In this way, solving the stochastic program with the scenario set should yield improved solutions.

The first method of this type was proposed by [75]. In this approach, for a pool of first-stage solutions, one minimizes a weighted sum of squared differences between the in-sample and out-of-sample values. The approach can simultaneously find new scenarios and probabilities for constructing a scenario set, but the regression problem is non-linear and non-convex and so is difficult to solve. The tractability of the approach depends on how complex is the recourse function for the problem.

The more recent papers [65, 89] also propose regression-based approaches but limit themselves to scenario reduction, that is, selecting a subset of scenarios from a larger set. In other words, we assume here that the true distribution is a discrete with a large number of outcomes. In the case of [89], a mixed integer linear program is formulated, which selects a subset of scenarios of given cardinality with new probabilities in such a way to minimize the total absolute error between in-sample and out-of-sample values. The authors identify that a potential problem for this approach is that preserving expectation for a sufficiently large pool of solutions may lead to a prohibitively large reduction problem. To address this, they propose a filtering approach that removes first-stage solutions whose behavior with respect to the objective function is similar.

In [65], rather than preserve expected values of the loss function directly, singular value decomposition is performed on a matrix consisting of loss function evaluations over the pool of first-stage decisions and all scenarios. This decomposition results in a set of singular vectors and values, and the magnitude of the singular values tells us how important each singular vector is in reconstructing the loss functions over our solution pool. Scenario reduction is then carried out by solving a linear program that selects a subset of scenarios and probabilities in such a way that preserves the expectations of the singular vectors with the largest singular values. The error between in-sample and out-of-sample values in this case can be bounded by a function of the singular values.

In all three of these approaches, one requires the construction of a solution pool. While [89] use single-scenario solutions (i.e., first-stage solutions found by solving

the stochastic program with only one scenario), [65, 75] advocate the use of better, more "relevant" solutions here. For example, these may be found by solving the stochastic program exactly with a small number of scenarios, or solving for a larger scenario set using a heuristic. Although the approach of [75] is more general than the latter two in the sense that it can construct new scenarios, it is also more difficult to apply in practice as the regression problem is more complex and may require tailored algorithms to solve. On the other hand, for [65, 89], given a set of first-stage solutions, no tailored algorithms are required. One essentially just needs to be able to evaluate the recourse function for different scenarios and first-stage solutions. In this way, these two approaches are fairly straightforward to apply. In all these approaches, problem constraints are only exploited indirectly through the generation of the solution pool.

Clustering Approaches and Problem-Based Probability Metrics

For distribution-based approaches to scenario generation, in clustering approaches, one groups together scenarios based on how close they are to each and then constructs a scenario set by selecting a representative scenario for each cluster. Clustering can also be used as problem-driven approach by instead clustering scenarios based on how they behave with respect to the stochastic program. An early example of such an approach is [24] who present a framework for problem-based clustering whereby for a given solution we define for each scenario a set of *sensitivity indices*, and then cluster scenarios based on these. The paper shows how this can be successfully done for a unit commitment problem; however, to apply this approach to other problems, one would have to find other appropriate sensitivity indices.

More recently, three other problem-driven clustering approaches have been proposed in [7, 37, 52] that use single-scenario solutions. This makes the approaches somewhat more applicable compared to that of [24], since they need less tailoring to work for different problems. The idea of the method in [52] is to cluster scenarios in such a way which minimizes the total *discrepancy* over all clusters. By discrepancy here, we mean the maximal error of using a representative scenario for a cluster over a solution pool. When this pool consists of the set of single-scenario solutions, this problem can be approximated by a mixed integer linear program.

Reference [37] define a distance metric between two scenarios based on the opportunity cost—that is the cost of using the optimal solution with one scenario for a different scenario. Scenarios are then clustered using this problem-based distance measure with graph-clustering techniques. Reference [7] also use this opportunity cost-based distance but embed it into a Wasserstein-like probability metric. Scenario sets are then constructed in such a way to minimize this new probability metric between the original scenario set and the new one. Unlike the other two approaches here, this approach can construct new scenarios that were not in the original set. On the other hand, this minimization problem is more difficult to solve, and so the method is tractable for a more limited set of problems.

In [36], the authors propose another probability metric that can used for scenario generation. Unlike the other approaches, the formulation of this metric depends

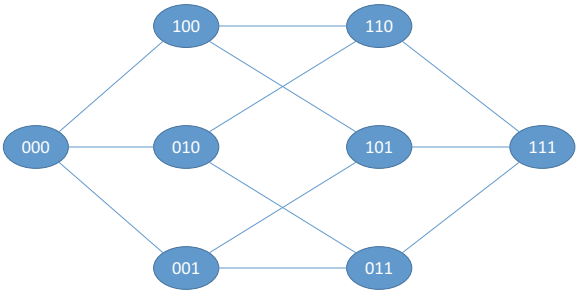


Fig. 4.8 Partial order of scenarios for three edges that can break down. For example, 010 means that edges 1 and 3 are broken, edge 2 operating. The lines show the partial order

on the whole set of feasible decisions, rather than a finite pool of decisions. The minimization of this distance is reformulated as a generalized semi-infinite program, for which numerical solution procedures exist. However, due to a lack of numerical experiments using this technique, it is not yet clear how tractable this method is in practice.

Other Problem-Specific Approaches

For many of the approaches described above, the scenario generation methods are applicable in principle to wide classes of stochastic programs. However, some approaches rely on much more specific problem properties.

An example of this type of method can be found in [74]. To illustrate this, imagine you have a two-stage problem, where the second-stage problem is a network flow problem with random edge breakdowns. The first-stage decision is maybe to set production or production capacities at a number of factories, and the second stage to meet deterministic demand at some demand nodes using the production (capacities) from Stage 1 and the functioning edges. Assume that if demand cannot be met, there is a penalty to be paid. Now assume that you have an edge breakdown scenario s that, for a given first-stage solution, results in a penalty for non-delivery. Then every scenario where the same edges, plus some more, are also broken will also lead to a penalty. Similarly, if a scenario s' does not lead to a penalty, then also all scenarios with the same edges, plus a few more, not broken, will also lead to no penalty. In the cited paper, it is shown how this can be used to create scenarios, or to simplify calculations in the second stage. This logic also applied to other tail risk measures. For an example, see Fig. 4.8 for a case with three edges. Let, for example, (101) imply that edge 2 is broken and edges 1 and 3 still operate. If any of these nodes produce a penalty for a given first-stage decision, then so will all nodes that can be reached by following lines to the right also produce penalties. Similarly, if any node does not lead to a penalty, the same will happen to all nodes that can be reached by following lines to the left.

There are also cases where specific problem properties are used to find good scenarios. In [90], for example, a simple three-point distribution is developed that does not try to mimic the real distribution at all. Its only goal is to produce a first-stage solution that "looks right." The in-sample objective function values

are meaningless, and all solutions are evaluated out of sample. So the idea is to understand what first-stage solutions will look like and build from there.

This might sound vague—but you could of course read the paper—that might help. To be concrete, however, in the next chapter we explain in rather great detail how the structure of first-stage solutions can be used for scenario generation for problems with a very high number of dependent random variables. That chapter helps you see how solution structures can be used.

4.6 What Approach to Pick?

So how should you go about this? If you are going to solve a certain problem only once, trying SAA first is probably a good choice. You may have to solve your model quite a number of times, but most likely that is the easiest way forward. But by now you understand that this might lead nowhere if the number of scenarios you can handle numerically is too low for the problem you solve to have any reasonable relationship to what you want to solve. This is the normal difficulty of sampling.

But if the model is to be solved repeatedly, the conclusion is not so obvious even if SAA generates reasonable results. If this is a model of the type that is solved every morning, say a production model, you are probably rather constrained on elapsed time. That may in its own right rule out SAA if you at the same time really care about the quality of the solution. If you instead use for example moment matching and establish in- and out-of-sample stability (and maybe even use one of the statistical approaches to get a feeling for statistical quality, but do it off-line), you can then proceed day-by-day by having to solve just one stochastic program. For sure this is a heuristic, but not one without any foundation in reality.

If you are in an academic setting and need to solve hundreds or thousands of similar problems to prove, for example, some properties of the solutions or solution procedures, which is often done in academic work, you would also hesitate on using SAA as each such case would then require a repeated solve of your stochastic program to establish statistical convergence. The elapsed time would easily be enormous. We are then assuming, of course, that you care that the problems you solve make any sense. So instead we would suggest that you spend a lot of time up front on figuring out how to generate scenarios so as to obtain stability, thereby reducing the solution times of the repeated cases substantially.

Chapter 5
High-Dimensional Dependent Randomness

With Zhaoxia Guo and Michal Kaut

With uncertainty, there is never a dull moment.
– *Haresh Sippy*

Risk means "shit happens" or "good luck"
– *Toba Beta*

The methodology outlined in Chap. 4 will need to be extended in many directions, and most of what is needed is so far not done. So here are openings for many interesting future theses and papers. In this chapter we shall show one possible path to take for some types of problems. Maybe this is useful for what you need. If not, maybe it can help you think about what needs to be done with your problem. Most of this chapter is based on the work in [30].

Assume you are facing a problem where the number of random variables is really large (tens of thousands) and that they obviously are dependent, and you suspect that the dependence matters (whether or not dependence actually matters is probably a numerical question). Let us first give you an example where this occurs.

Consider a problem from city logistics. The problem itself is a classical vehicle routing problem. You have a number of trucks, a number of customers with given deterministic demands (we leave random demands out for now), and possibly other attributes such as time windows. As usual, traffic conditions are not known. You may know that there are daily patterns such as morning and afternoon rush hours (known as time-dependent travel times in the literature), but in addition there is a lot of uncertainty in traffic conditions, caused by weather, accidents, closed tunnels, and bridges, who knows … The goal is to find routes for the trucks so as to visit all customers, obeying the vehicle capacities, while minimizing expected overtime pay for drivers. Of course, there are many possible objective functions that depend on total travel times, and we picked expected overtime pay as it is simple to understand. So the drivers have a normal working day. If they get back early, we save nothing, but if they are back late, we must pay them extra. This

A. J. King, S. W. Wallace, *Modeling with Stochastic Programming*, Springer Series in Operations Research and Financial Engineering,
https://doi.org/10.1007/978-3-031-54550-4_5

choice is not crucial for what we discuss in this chapter; we just need something that depends on the distribution of travel times and preferably something that is not symmetric like expected travel time. The point of the "symmetry argument" is often valid in stochastic programming. If being 5 minutes early cancels out being 5 minutes late (which is implied by, for example, minimizing expected travel time), then deterministic solutions are often rather good as stochastic effects cancel out to a large extent. So we use a non-symmetric objective function in our tests. If you solve a real problem, you will of course use whatever is the right objective function in your setting.

We shall talk about stochastic speed rather than stochastic travel times. The reason is that if we operate with, say, 10-minute time intervals, it is more natural to say that speed changes every 10 minutes than that travel times change every 10 minutes. Most likely a trip will cover several time periods, and the travel time is a result of many different speeds.

Hence, since speeds vary over the day, we shall need to split the day into discrete time periods. A road link has different expected (and different realized) speeds in different time periods. Moreover, speeds along a given link may be dependent, in a probabilistic sense, on speeds along other links. Speeds—traffic conditions—on links leading into or out of an intersection will be dependent. If there is trouble on one link, there is probably (but not for sure) trouble also on the other links getting into or out of the intersection. And with time discretization, the speed on the same link in neighboring time periods is positively correlated (so dependent); the more so the shorter the periods. This seems reasonable from a modeling perspective, and it is confirmed by data.

So what is the point of modeling random speeds? First note that deterministic time dependence takes care of some important issues, for example, avoiding rush hours if that is at all feasible. And even more importantly, it helps us find routes that avoid the worst effects of rush hours by traveling against traffic if that makes sense in our setting, for example, going into the city center when the major traffic goes out and vice versa. And of course, deterministic dependence can make sure that the areas where the rush effects are worse are avoided (if feasible) at those times. If you have one truck and a small city, this can probably be done by hand, but as the problem size increases, optimization is needed; but based on what we just outlined, a deterministic model will possibly do.

So what more do we need that deterministic time dependence does not give us? The simplest answer is, in reality, there is a trade-off between expected travel time and its variance. A deterministic method would try to fill the available time for drivers, resulting in very high probabilities of overtime pay. By including stochastics, we shall see that trade-off by having expected route travel times slightly below the available time. Furthermore, since most travel time correlations are positive, the optimal solution will look for routes where the positive correlations are small, as that reduces the variance in overall route travel time for a given mean travel time. For other objective functions than the one we use, it may also be important to understand correlations between routes so that, for example, not all routes are late at the same time.

It might seem that an intermediate solution between deterministic speeds and full dependence is to let speeds be random but independent. There are some papers that do that. First, it is not necessarily easier to do that. If you generate scenarios by sampling or some other approach like we outlined in Chap. 4, ensuring uncorrelation is as hard as ensuring any other correlations. However, there are approaches that utilize the relative easiness of adding independent distributions. It is easy to calculate the time needed to travel along a route from the depot to a number of customers and back in the independent case. But think about it. Time independence means that zero speed in one period can be followed by free float traffic in the very next period. This makes no real sense, and it is hard to see (but feel free to check) that it is any better than time-dependent deterministic speeds. This type of independence is far too "nice." It will allow low and high speeds to cancel out, while in reality (almost) all speeds are positively correlated, so that when you face trouble on one link, you are most likely facing many troublesome links; you have a bad day. With positively correlated speeds, the variance of travel time on a path is much higher than in the uncorrelated case. Negative correlations would make the travel time variance even smaller, but we do not face that in most traffic problems. Make sure you understand this logic.

So the alternative to deterministic time-dependent speeds seems to be stochastic and dependent. As mentioned, although all (most) speeds are positively correlated, the correlations are rather different. And a large positive correlation is more troublesome than a small one. So, based on the objective function, an optimal solution would trade off expected route lengths of a truck with its variance (or some other statistical property), to end up with the optimal route travel time distribution.

5.1 The Stages

The above general argument covers many different problem formulations. The issue is, as always, what we know and when do we know it: the stage structure. Depending on the real setting, the first stage could simply be where to go first (the rest to be decided later; a multistage formulation), the whole route of each vehicle (so the second stage is just to calculate the travel time distribution and calculate the objective function value), which customers to visit by a vehicle (but not the order; most likely a multistage formulation), or the sequence of customers (but not the actual routes; could be both two-stage and multistage). We shall have to make some assumptions now to illustrate our ideas.

Firstly, think of the driver when he is about to leave a customer and has decided where to go next (whether that was a first-stage decision or something he just decided). Assume he has a satnav in the vehicle and that it uses present speeds on all links (real-time information on traffic conditions) to calculate the quickest route to the next customer. He then follows that route even if speeds change. We shall also allow for the possibility that the satnav changes "its" mind whenever speed information changes, implying a rolling horizon view of the problem. But

we shall not cover a truly dynamic setup where the future changes are anticipated. That would require a multistage formulation, which would be nice, but must be left for later (for you?). So we shall have the "correct" travel time distribution on the path, but not a true multistage formulation of the path choices. The main reason for making this choice is that satnavs do not anticipate later choices even if they re-optimize whenever they have new information.

So our setting will be a driver who is being told which customers to visit and the sequence and who lets the satnav pick paths for him, using real-time speeds. Even though there are other meaningful setups, we feel this is a reasonable one—hopefully you agree.

5.1.1 Stage 1

So Stage 1 is to find the routes for all trucks, where a route is understood as a sequence of customers. This is the same as in any deterministic vehicle routing problem, and if heuristics are used, potentially heuristics from the deterministic problems still apply as they are all about customer sequences.

5.1.2 The Random Variables

So far we have described the random speeds on the road links, but the first-stage decisions only concern the nodes where there are customers. As you probably know, deterministic vehicle routing problems normally operate on the network of customer nodes, not on the network of all links—the map. In the deterministic case, going from the map to the customer node network is easy (and what people do); for each node pair you calculate the shortest (quickest) path on the map and use that as the distance between the nodes in the customer node network. But for the stochastic case it is much more difficult. If there was only one time period, scenarios on link speeds could be transferred to scenarios on node pair travel times, and then you could operate on the customer node network. However, in this case, even if link travel times were independent, the node pair travel times would be dependent as many pairs would share links. So assuming that node pair travel times are independent is even stronger than assuming that link travel times are, as it means that the different node pairs will observe different travel times on the same link. In our view that makes no sense at all.

> Formulating stochastic vehicle routing problems on the customer node network makes little sense. It is necessary to operate on the map.

With several time periods it gets worse; the quickest route between a pair of customer nodes will depend on not only which period you leave the first customer but also when in the period you leave, as that affects how much time you spend with each speed along the route. So going from scenarios on speeds in (link, period) pairs to (customer pair, period) would not be possible as long as trips do not start and end in the same period. In other words, we operate on the map, and a scenario contains a speed for each link in each time period, so the dimension is potentially very high.

5.1.3 Stage 2

From Stage 1 we have sequences of customers (one for each route) and from the random variables a set of scenarios, where each scenario covers all links in all time periods. What we therefore need to do is to find the shortest path between each pair of customers on each route for each scenario. In other words, what needs to be done on a single scenario corresponds to a sequence of standard deterministic shortest path computation. We use Dijkstra's algorithm, based on the speeds at the time when the truck leaves the first customer in a customer pair. We then calculate the actual travel time based on the scenario we are on.

As we see, the calculations in Stage 1 and Stage 2 are quite similar to what we would do in a deterministic case. The only issue is that Stage 2 must be carried out for each scenario, and hence, the workload will primarily hinge on how many scenarios we need to get good solutions, i.e., for stability.

5.2 Guessing a Correlation Matrix

Let us take a small sidestep and think about what we often do in optimization, particularly if we are interested in algorithmic developments. Unless there are standard test sets of problems that "everybody" uses, we resort to fake problems, problems we have made up ourselves. Often the paper has a section on how test cases were created. This is accepted in the scientific community. Let us assume that we are in that situation, and that we want to create a test case for a situation with dependent random variables. In such a case, it is hard to see how we could avoid having to come up with a correlation matrix, as that is, in a sense, a minimal requirement to represent dependence. We will then face a problem we are not used to: To guess a correlation matrix, especially in high dimensions, is basically impossible (except for the uncorrelated case). You need to come up with a positive semi-definite matrix (all eigenvalues must be non-negative). You may be lucky, but that is not very likely: Just try! This is true whatever stochastic program you want to solve: Guessing a correlation matrix is close to impossible (and even worse if you want it to make any sense).

This problem is general, and will face us also in the case we are looking at in this chapter: If we don't have any data, and just want to test our method with dependent random variables, we are in trouble.

It might be easy to find a reasonable matrix that is almost positive semi-definite. After all, we know what the correlations mean, for example the correlation between speeds on a given link in two consecutive periods. But to get it totally right is very hard. There are methods to find a positive semi-definite matrix as close as possible to your favorite matrix, but that will cause strange correlations here and there. You may end up with some large negative correlations in strange places—so be warned. It will not be easy to interpret the results afterwards.

And in case you think about sampling scenarios, we ask: From what?

5.3 The High-Dimension Hits Us

Let us assume that we have data for speeds, possibly captured via traffic management systems, or possibly even taxi data (yes, we have that). And let us disregard the statistical question of whether or not we have enough data points to end up with a trustworthy estimate of the speed distribution. This statistical question is indeed important, but outside the scope of this book. So now we have N data points of the speed distribution, where each data point has a huge number of outcomes, namely T (the number of time periods) times K (the number of links in the network), giving us $n = T * K$ random variables. In our largest case, $n = 25{,}080$. With data we can certainly calculate a correlation matrix, but it has over 310 million distinct correlations, so handling it will not be easy. But it is certainly doable. And in case you wonder it will certainly be positive semi-definite. There is never a problem with that when the matrix is calculated from data. With the data points, we can also easily calculate empirical marginal distributions and certainly expected values.

Expectations, variances, and a correlation matrix describe a Normal distribution uniquely. If the distribution is anything but Normal, what we have is simply an approximation. It is very likely that your marginal distributions contradict the idea of a multi-Normal. But let us not worry about that now (something for you to think about, maybe). So if you insist on sampling your scenarios, you can now do it, either from your empirical distribution or from your multi-Normal. You will notice, if you try, that sampling in extremely high dimensions is far from simple, numerically speaking, even though it poses no problem for the theoretical discussion.

If we now go back to Chap. 4, we will see that the methods outlined there do not lend themselves to scenario generation in this kind of dimensions. A correlation matrix of dimension 25,080 by 25,080 is not easy to handle, and from a city perspective, this is a rather small case. We would claim that apart from sampling, none of the existing methods would work. And in these kinds of dimensions, sampling would most likely lead to a huge number of scenarios—far too big in fact, for anything useful to happen if we have any requirements on the quality of

the solutions. Of course, challenging this view is indeed encouraged and can lead to many interesting papers.

So, that brings us to this chapter: What can we do? We should be able to do something, since most (not all) real problems are much bigger than what is being solved in the literature. We are sure this question of handling huge dimensions will produce papers and theses over the next years. In this chapter we will point in one possible direction, primarily to say that: Yes, we can solve huge problems. Then it is up to you to do it even better, by making our approach even better or by coming up with much stronger ideas.

5.4 Problem Structure

The approach we are going to use depends on understanding the structure of (optimal) solutions to whatever problem you are studying—in our case vehicle routing. So this falls into the category of problem-driven scenario generation. For this case the optimal solution is a set of routes (obviously). A route in our time-space network consists of a series of link–time pairs: You start at the depot, and then follow a link (road) either until the time period is over or until you reach an intersection, whichever comes first. You then move to either the same link in the next period or a neighboring link in the same period. This continues until you are back to the depot.

At this point in time, let us remind ourselves that there is a paper by Kaut [48] that allows us to create scenarios based on marginal distributions and a collection of bi-variate copulas. The method will try to match the marginal distributions and the bi-variate copulas, if at all possible, in the scenarios. The more scenarios we allow, the more freedom we have to match the properties. If copulas mean nothing to you, think of this as matching the marginal distributions and a subset of correlations. For normal distributions, the two are equivalent.

Given the structure of feasible solutions—routes—we shall use the following bi-variate copulas. For each time–link pair, specify the bi-variate copula relative to the same link in the two neighboring periods and to the links that it shares a node with in each period. In a typical Manhattan-like network, for a given link we would then specify two bi-variate copulas for the same link in the neighboring periods, plus copulas for seven links in each end node (with four two-way roads in an intersection, where each direction is counted as a link, there are seven other links). That adds up to 16 copulas. As all of these will be counted twice, we have a total of eight defined bi-variate copulas per random variable. And this number is independent of the size of the network. This is extremely low compared to the number of bi-variate copulas that involves this time–link random variable, which is 25,079 per random variable (or half that for the same double-counting reason).

So we shall generate scenarios that try (and mostly succeed) to match the marginal distributions and the given bi-variate copulas. If the match was perfect, our only potential problem when calculating the distribution of route travel time

would be that some time–link pairs on the routes, for which the bi-variate copula was not specified, might have (and normally will have) correlations that are off in the scenarios. What kind of trouble can that cause? Well, in its own right, that is hard to say. But it will for sure create errors in the distribution of travel times on the routes, but how much does it matter? That, again, depends on the objective function. Note that if the method does not perfectly match the specifications, it will also cause some potential trouble.

But luckily, we have a tool for this, namely our stability tests from Chap. 4. If the scenarios generated by using a small subset of the bi-variate copulas (correlations) produce in- and out-of-sample stability at an acceptable level, we know we have OK results, and that the incorrect correlations did not matter too much. This is true even if we cannot fully describe exactly what happens. And our tests show that in dimension 25,080, we need in the order of 15 scenarios to obtain an error of less than 1% for the most objective functions. We believe that to be a very low number for such a high dimension. It will mean that a function evaluation in the stochastic case would take about 15 times longer than in the deterministic case. It is worth noting that as the number of scenarios is increased, the error goes down, but apparently not all the way to zero; incorrect correlations will always have some effects. But we believe that for most problems, an accuracy much below 1% makes little sense in any case, given the quality of other numbers and modeling assumptions.

5.4.1 So Why Does This Work?

As you can see, we could solve, approximately, problems with over 25,000 correlated random variables. Why did it work? We would say because we understood which correlations (bi-variate copulas) were most important, and this understanding was problem-dependent. Furthermore, the stability tests helped us show that the setup worked even though we had potential numerical problems. We believe that you can approach other problems in high dimensions in a similar way using the same scenario generation method based on the structure of solutions. Good luck!

Chapter 6
Multistage Models

> There is stability in walking an uncertain path, because you never allow yourself to be misled by what you think you know.
> – *A.J. Darkholme, Rise of the Morningstar.*

In the previous chapters we focus on two-stage models, for many good reasons. First, the size of the problem is exponential *in the number of stages*. Second, it is hard enough to model one stage of uncertainty. Why add to the difficulty by having multiple stages?

Remember something we said earlier in the book. Stochastic programming is generally about transient decision-making. We care about what to do *now*, not what to do in all possible "nows." And to decide what to do now, we need to understand the consequences of what we do now in light of an uncertain future. To a large extent, it is the future (of course not just the uncertainty of the future) that we make decisions for, though there might certainly also be immediate costs and incomes. So the stages in a model are there to capture the effects of what we do now, and they are *not* there for us to plan for later decisions. This is a central observation:

> The purpose of the stage structure and all decisions except those in the first stage is to capture the effects of the first-stage decisions and send a message back to the first stage about these effects.

This understanding is crucial when creating multistage models. It has some profound consequences. The first is that the quality of a stochastic program cannot be determined by looking at how well the stages actually describe the problem. We shall see an example in Chap. 7, where an inherently two-stage problem—network design—with infinitely many stages is modeled as a two-stage stochastic program, and where the second stage is clearly *wrong* but still sends good information back to

A. J. King, S. W. Wallace, *Modeling with Stochastic Programming*, Springer Series in Operations Research and Financial Engineering,
https://doi.org/10.1007/978-3-031-54550-4_6

Stage 1. So though it is totally fine to model the later stages as correctly as possible, that is in its own right not the point of later stages. If you limit your modeling by thinking about the correctness of the problem description, rather than sending of information back to Stage 1, you may miss many opportunities for a functioning model.

So we do not model many stages just because the problem can be described with a long, maybe infinitely long, sequence of decisions, learning, decisions, learning ... Rather, we do it because the consequences of a decision simply cannot be captured in a two-stage model. Here are a few cases:

- An option, be it real or financial, that can be exercised at several different points in time, and where this flexibility in time is (or is expected to be) important for its value, will probably need at least three stages to capture the value of the flexibility. Otherwise it will not look more valuable than the same option that can only be exercised once.
- An option, be it real or financial, which can be bought now, later, or never, most likely needs at least three stages to capture the value of possibly waiting. In a two-stage model the question ends up being "now or never" and that misses the value of waiting to see what happens.
- The first stage in an inherently two-stage problem may in fact have several stages before the usage part takes over. An example would be that you are planning to set up a network of warehouses. First you decide on locations. These may have to be bought, and that takes time. Then at a later point in time you decide exactly the size and properties of these warehouses. In the mean time you have learned more about the market in which you are operating. So the model has at least three stages, the third being the use of the facilities.
- If the inherent second stage is infinitely (or very) long, we might use ideas from Sect. 6.6 to make it finite. However, very often the result is that we need more than just one time period (stage) before we add the horizon effect. If so, we will end up with a small number of stages, but such that the model is multistage.

6.1 Capacity Planning Option Example

In the rest of this chapter, we will work through the formulation of a canonical example in stochastic programming called the capacity planning option problem. In this problem, the main uncertainty comes from future demand. The demand can be met by production from the facility, up to the capacity limit, or by drawing down the inventory. At some point in the future, there may be a need to increase capacity. This decision, of course, depends on the profitability of adding additional capacity.

As is often the case in planning for future events, there may be an opportunity to purchase an option today that gives the right, but not the obligation, to add additional capacity at some future dates. In well-developed commodity markets, such options

are available with standard terms and conditions; purchasing and exercising such options are purely financial transactions. But there may be other types of options available. For example, the investor may need to decide today whether to purchase land next to the existing production facility, and perhaps the decision needs to be made soon because the land is for sale. If the land is purchased now, then perhaps adding capacity in the future will be much cheaper. Indeed, the entire point of building this capacity planning model could well be the evaluation of this kind of option.

As we just noted, a decision about an option may require at least three stages. Let us keep this in mind as we model the problem. We begin with the capacity model that describes the option. Then we develop the inventory model that connects capacity to the uncertain demand, and finally we discuss how to organize the time evolution of the variables using the concept of information states and construct the timeline of observations and decisions.

The convention in multistage is to label "today" as a Stage 0, so the "here and now" decisions are made in Stage 0. We imagine Stage 0 to represent an instant of time. But future stages beyond Stage 0 represent intervals of time, like a day, week, or year. The intervals do not have to be all the same length.

6.1.1 Capacity Model

The initial capacity K_0 is known at time 0. There is additional optional capacity of size K_Z available, but the investor does not want to use it now. Instead, the investor wants to buy an option to use it in the future. The option has an expiration date T_Z, and the option can be exercised at any date up to and including T_Z.

We model the option with a sequence of binary variables, $z_t \in \{0, 1\}, t = 0, \ldots, T_Z$. The first, z_0, is the decision to buy the option, and the subsequent variables model the decision to exercise the option. The capacity starts with K_0 and jumps up by K_Z when the option is exercised. Here is the capacity model, along with a constraint that describes the option exercise

$$\begin{aligned} K_t - K_0 &= z_t K_Z \;\; t = 1, \ldots, T_Z \\ \textstyle\sum_{t=1}^{T_Z} z_t &\le z_0 \\ z_t &\in \{0, 1\} \; t = 0, \ldots, T_Z \end{aligned} \tag{6.1}$$

Can you see the way this works? It says two things: First, the possibility to exercise is available only if z_0 is non-zero, and second, the exercise happens only once. We plan only for one exercise time in this example, but the model can be extended to any number of options.

6.1.2 Inventory Model

The simplest model for inventory is: The change of inventory equals production minus demand

$$\begin{aligned} I_t - I_{t-1} &= x_t - d_t \quad && t = 1, \ldots \\ I_t &\geq 0 && t = 0, \ldots \\ x_t &\in [\,0, K_{t-1}\,] && t = 1, \ldots \end{aligned}$$

where inventory I_t is always non-negative, production x_t is non-negative and bounded above by capacity K_{t-1}, and the observed demand is d_t. The starting inventory is I_0.

Notice that this inventory model requires that production x_t plus existing inventory on hand I_{t-1} be large enough to satisfy demand. Obviously, if the results from the model are to be used in reality, a shortage may occur if real demand is larger than the largest one we put in. The model does not say anything about what will happen then. Hence, while demands must be met at any cost in the model, demands above that maximal level we put in do not matter at all! So the model is a bit shaky: On one hand we require feasibility at any cost, while at the same time, in reality, shortages might occur. This does not represent good modeling.

Ways around this could be to think seriously about modeling backorders or lost sales. But it is simpler to focus directly on the main issue: The model requires meeting demand, which we do not control and in reality will be something we do not know in advance.

Perhaps we can change our perspective on demand. Let us view it more like a *potential*—it represents the potential for sales (since there is a customer) but not actual sales (since we may not have supply). Let us create a sales variable y_t with demand as the upper bound and revise the inventory equations

$$\begin{aligned} I_t - I_{t-1} &= x_t - y_t \quad && t = 1, \ldots \\ I_t &\geq 0 && t = 0, \ldots \\ x_t &\in [\,0, K_{t-1}\,] && t = 1, \ldots \\ y_t &\in [\,0, d_t\,] && t = 1, \ldots \end{aligned} \tag{6.2}$$

This allows demand to be higher than what can be obtained from the supply ($I_{t-1} + x_t$) and also incorporates the actual input and output flows into the inventory stock model. We think this is a reasonable solution for this simple case.

Let us consider the sales variable y_t just introduced. Is it really a "decision"? To analyze this let us consider the aspects of the model related to sales. First, we cannot sell anything unless there is demand. Second, the supply is production plus the stock of inventory: $x_t + I_{t-1}$. We cannot sell anything unless there is supply. Since sales has to be less than the minimum of demand and supply, we can say

$$y_t \leq \min\{d_t, x_t + I_{t-1}\}$$

Now here is the question: Why would the business ever decide to delay a sale? That is what it would mean if y_t was strictly less than the minimum of demand and supply in t. If demand was always just for one widget at a time, this would make no sense. A sale tomorrow is the same as a sale today in our deterministic situation. Furthermore, when future demand is uncertain, a sale today is much better than an uncertain sale tomorrow.

A very good reason to refuse a sale today, on the other hand, is that we have an opportunity to bid on a single large order for future delivery. This may fetch a higher price because it is in effect reserving future capacity. Even if the price is the same, it may be worth accepting the order to guarantee utilization of the capacity. So it might make sense to plan enough inventory to fill that future order. This problem is called "lot sizing."

Modeling and solving lot-sizing problems are very important. It is the real inventory problem faced by most businesses! The big challenge with the lot-sizing problem is assessing the future demand for orders of a particular size and delivery schedule. If we decide to model lot-size orders, we will also need to separate the lot-size demand data from the lot-size sales decisions, as we have done above. The technical complications for lot sizing are very great. The literature on this topic is very deep and interesting, but we think it is beyond the scope of this chapter. For further reading on stochastic lot-sizing problems, see the recent papers [10] and [11].

6.1.3 Objective

The natural objective is to maximize the reward: revenue from sales minus costs of operation. Here are the definitions:

$$\begin{array}{lll} \text{Revenue from sales:} & r_t y_t & t \geq 1 \\ \text{Cost of production:} & c_t x_t & t \geq 1 \\ \text{Cost of storage:} & f_t I_{t-1} & t \geq 1 \\ \text{Reward:} & R_t = r_t y_t - c_t x_t - f_t I_{t-1} & t \geq 1 \end{array} \tag{6.3}$$

Next, there are the costs of option purchase and exercise:

$$\begin{array}{ll} \text{Cost of option purchase:} & C_0 = g_0 z_0 \\ \text{Cost of option exercise:} & C_t = h_t z_t \quad 1 \leq t \leq T_Z \end{array} \tag{6.4}$$

With these definitions, we may write the initial draft of the capacity option problem as

$$\max_{x,y,I,z} \sum_{t \geq 1} [\, R_t - C_t \,] - C_0 \tag{6.5}$$

subject to the capacity model Eqs. (6.1) and the inventory model Eqs. (6.2).

The objective as written assumes that a payment far in the future is evaluated with the same weight as a payment today. So should we consider discounting the future payments?

This is not a simple question. Businesses may have all sorts of uses for the cash flows from earnings. For example, the objective model does not include any of the details from other expenses of running the business: Perhaps there is a large loan repayment in the future. On the other hand, the production-inventory model may apply to one business out of many. There may be an aggregate model for these larger questions. The output of this production-inventory model may be used to feed information into the aggregate model, in which case only the cash flows are required. In fact, this is what we do in Sect. 6.8.2 when completing the capacity option model.

As a general rule, it is better not to worry about discounting until these issues are better defined. The major reason for discounting is to model the horizon, either as an infinite but gradually less important influence on current decisions or as part of a procedure to compress the infinite future into a single horizon stage, as discussed in Sect. 6.6.

6.2 Mapping Dynamics into Stages

Stochastic programming models focus on the first-stage decision. In the model we are building, the first-stage decision is whether or not to purchase an option to install additional production capacity. The impact of additional production capacity is measured by the profit generated by the linked dynamics equations for capacity (6.1) and inventory (6.2). The timing of the option exercise decision is uncertain because the value of exercise depends on future profits. Even though we focus on the option purchase decision, we require multiple stages to compute the risk of this decision.

This section will focus on the modeling choices for mapping the dynamics equations into a multistage stochastic program. Just like in two-stage stochastic programming, the elapsed time between stages is a modeling choice. Multistage stochastic programs introduce some additional modeling choices for elapsed time between stages.

6.2.1 Timeline

When designing stages, it is always a good idea to understand the data recording procedures. For example, sales data may consist of time-stamped order arrivals and sales deliveries, inventory data may be collected daily or monthly, and the option exercise decisions may be scheduled quarterly. The stochastic programming point

of view is that the recording of data is an event: It is the time or date when a piece of information is observed. The modeling challenge is to map these event sequences into a timeline.

> You cannot set up a meaningful stochastic program without a proper timeline.

The dynamics equations describe the logical order of the events. Demand arrives. If there is enough product, we can make a delivery and record a sale. But when event sequences are mapped into a timeline, this logic may no longer apply.

To illustrate the issues, let us consider making a stage of length 1 week. The natural way to group time-stamped transactions into into a stage of length 1 week is to aggregate them, so the weekly orders and deliveries are the sums over the demand observations and sales decisions made during that week. But now the natural logic of a single order followed by the delivery of that order no longer applies. When we group flow variables together into a single stage, the natural order of the events is mixed up.

The major modeling issue when grouping sequential events into stages is to decide which comes first: the decision or the observation. There are two choices:

1. Uncertain events are observed *before* decisions are made.
2. Uncertain events are observed *after* decisions are made.

Neither of these is better than the other. But we need to make a choice. A good way to proceed is to think about two stage:

- *Observe* the initial conditions.
- *Decide* the first-stage decision.
- *Observe* the random event.
- *Decide* the recourse decision.

The choice (1) fits this pattern. So let us agree that when aggregating events, we apply the *observe-decide* pattern.

In Table 6.1 we apply the *observe-decide* pattern to derive the timeline for the dynamics equations (6.1) and (6.2).

Table 6.1 Timeline for the production-inventory-option problem

Stage	$t = 0$	$t = 1$	$t = 2$	$\cdots$
Old State	–	(I_0, K_0)	(I_1, K_1)	$\cdots$
Parameters	K_Z, g_0	(c_1, r_1, h_1)	(c_2, r_2, h_2)	$\cdots$
Observation	(I_0, K_0)	d_1	d_2	$\cdots$
Decision	z_0	(x_1, y_1, z_1)	(x_2, y_2, z_2)	$\cdots$
Rewards	$-C_0$	$R_1 - C_1$	$R_2 - C_2$	$\cdots$
New State	(I_0, K_0)	(I_1, K_1)	(I_2, K_2)	$\cdots$

In addition to the categories of decisions and observations, we include three additional categories: the old state, the rewards, and the new state. The old state consists of the inventory and capacity variables computed from the stage before. The reward is the contribution to the objective value from the decisions and observations made during the stage. The new state becomes the old state for the following stage. We call capacity and inventory "state variables." This distinction is not emphasized much in stochastic programming, because not all problems have state variables as such. However, state variables are extremely useful when they exist. The dynamics equations in the production-inventory problem clearly identify capacity and inventory as state variables.

6.2.2 Information State

The timeline shows that the end of a stage is marked by the calculation of the new inventory and capacity states. The *observe-decide* pattern means that the observations of all events before the end of the stage are available to make all the decisions in that stage. The information available to the stage decisions constitute the *information state* of the stage.

> This information state consists of the models, processes, observations, actions, and rewards available to calculate the decisions.

Table 6.1 uses the mathematics of the stochastic program to define the objects needed for stage t next state calculation. The information state describes how to generate these objects.

Let us look at the stage $t = 2$ column in Table 6.1. The listed information available to compute the decisions (x_2, y_2, z_2) consists of:

- Past information states for stages $t = 0$ and $t = 1$
- Current observation d_2
- Reward function $R_2 - C_2$

But there is a lot that is missing. For example, how are the demand observations generated?

6.2.3 Recipes

It is useful to consider the information state as being produced by "recipes." These recipes list the ingredients and describes the processes and models that generate the

data used by the decisions and what updates to pass to the next stage. To get an idea for what may be involved, let us discuss the information state for the initial stage $t = 0$ and then the information state for stages $t > 0$.

6.2.3.1 Recipes for Initial Stage

The column $t = 0$ in the timeline table classifies the inventory and capacity states (I_0, K_0) both as observation and as final state. There is no mathematical formula for generating the inventory and capacity quantities. They are observed, not computed. But it is not enough to say "we observe something." We need to state how we are going to observe it. Which database? This is what we mean by recipe.

The information state is the container for the recipes. The title of the first recipe might be:

> *Recipe for observing* (I_0, K_0) *as required by the final state.*

Then it might list some database processes (tables, SQL code, and so forth). The title for the next recipe might be:

> *Recipe for determining the parameters* (K_Z, g_0) *as required by the first stage option purchase cost, and by the* $t > 0$ *capacity update equations.*

Collecting the information to generate these observations and parameters might not be a simple task! More importantly, for stochastic programmers, the implementation of the information state may reveal additional sources of uncertainty. The optional capacity K_Z might not be known. Perhaps there is an ongoing negotiation. Perhaps there are additional costs that we do not know yet. We may even decide to introduce an additional stage!

6.2.3.2 Recipes for Dynamic Stages

The information states for stages $t > 0$ are formed around the observations of random demand d_t. The first recipe in the information state might be:

> *Recipe for observing random demand, required by the inventory dynamics and the sales and production decisions.*

In reality all randomness in stochastic programming is generated from models such as those discussed in Chaps. 4 and 5. These models may require parameters; for example, the variance–covariance matrix may be required in a model that uses Gaussian copulas. The recipe has to specify what models and where the parameters come from.

Parameters may vary from stage to stage or may even be affected by external events like political decisions and changes in customer preferences. More importantly, as stochastic programmers, we may discover that the impact of these external

events are the most important features of the uncertainty. We may even decide to restructure the timeline and the information states to focus on the uncertainty models for these events!

The dynamics for our production-inventory problem are also part of the information state! The mathematical model does supply the inventory and capacity update equations, which are used to generate the final state (I_t, K_t). But bear with us for a moment. Let us just say that there needs to be a recipe:

> *Recipe for updating the inventory and capacity final state from the demand observations and the sales and production decisions.*

Then let us see why we want to include it. Well, in the future it is possible that the dynamics might change. For example, there may be new regulations that require changes to production processes. Perhaps it is uncertain exactly when these will be imposed. Or there may be a source of uncertainty between the production schedule and what is actually available for sale. Do you see why a recipe might be needed here? Are there other recipes that might be needed?

6.3 Modeling Multistage Uncertainty

The most complex part of multistage stochastic programming is the model for uncertainty. It is hard to avoid highly technical topics. The development in this section is centered around scenario trees.

Scenario trees are the most useful description of multistage uncertainty. You may have taken courses on stochastic processes, and you may have absorbed the mathematical myth that the most general description of uncertainty consists of continuous time processes with continuous distributions. At the risk of starting a controversy, we have to point out that this is emphatically just a myth. Continuous time and space models are useful mathematical abstractions that in special cases lead to useful closed-form solutions. For example, in options pricing the continuous time model is used for sensitivity analysis and hedging options with exercise dates that are weeks away. But the myth fails badly close to exercise date.

We do not intend to suggest that scenario tree technology presents a profound mathematical advance. It is just a data structure. We are only asserting that if you have to account for multiple stages in your decision problem, then use a scenario tree. Draw pictures of scenario trees. It will be a better guide for your intuition.

The main challenge in multistage uncertainty is not technical. The main challenge is modeling. Our models must be able to connect all those parameters together in a logical framework. Finally, as always in stochastic programming, our modeling focuses only on the quality of today's decision.

6.4 Scenario Tree

You have earlier, in Fig. 1.1, seen a (simple) scenario tree. That tree branched each time we learned something, and each node in the tree had its own data and decision variables. Chapters 4 and 5 discuss how such trees can be made. The branching from the root node to the nodes in time $t = 1$ is easy to visualize. But to work with multistage scenario trees, it helps to develop some notation.

The collection of scenarios is $\mathcal{S}$. Each scenario $s \in \mathcal{S}$ visits a sequence of information states, $n \in \mathcal{N}(s)$, which we call nodes. As discussed above, the information state represented by a node includes, in addition to the dynamics and the state transitions, information about the (discrete) probability distribution that generates the observations for the next state. Each point in the support of this distribution generates a transition to a new information state, to a new node.

Nodes represent information states, so there is a time stage $\tau(n)$ associated with each node n. Figure 6.1 illustrates the time stages on the right of the diagram; the convention in scenario trees is to place nodes with the same time stage at the same level. The subset of scenarios passing through the node is denoted $\mathcal{S}(n)$, and the collection of nodes at level t is denoted $\mathcal{N}_t$. Can you see that the nodes $n \in \mathcal{N}_t$ represent subsets of scenarios? The collection of subsets $\{\mathcal{S}(n), n \in \mathcal{N}_t\}$ is actually a *partition* of the scenario set $\mathcal{S}$. For this reason, $\mathcal{N}_t$ is called the "scenario partition" for time stage t. Let us summarize the notation used to relate scenarios, nodes, and time stages:

$\mathcal{S}(n)$	*set of scenarios passing through node n*
$\mathcal{N}(s)$	*set of nodes visited by scenario s*
$\tau(n)$	*stage of a node n*
$\mathcal{N}_t$	*scenario partition for time-stage t*

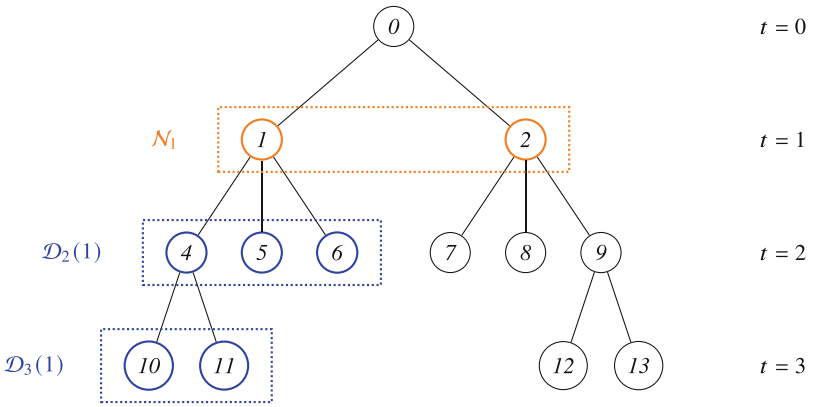


Fig. 6.1 Illustration of scenario tree highlighting scenario partition $\mathcal{N}_1$ and descendants $\mathcal{D}_2(1)$ and $\mathcal{D}_3(1)$. Note that the scenarios do not all have the same number of nodes

The defining property of a tree is that every node (except the root node) has one and only one parent, so the sequence of nodes leading back from a node n to the root is the same for every scenario in $\mathcal{S}(n)$. This means that every node n in the scenario tree is itself a kind of root for the sub-tree of scenarios $\mathcal{S}(n)$ that pass through that node. Occasionally, we will have a need to refer to nodes in the sub-tree $\mathcal{S}(n)$ with time stage $t > \tau(n)$. These are the "descendants" of n:

$$\mathcal{D}_t(n) = \bigcup_{s \in \mathcal{S}(n)} \mathcal{N}(s) \cap \mathcal{N}_t \tag{6.6}$$

As we proceed with the discussion, we will also find it convenient to use notation for parent and child nodes:

$$\begin{array}{ll} n^- & \textit{the parent of node } n, \textit{with time-stage } \tau(n) - 1 \\ n^+ & \textit{the children of node } n, \textit{with time-stage } \tau(n) + 1 \end{array}$$

Figure 6.1 shows a scenario tree with four stages. How many scenarios go through node 0? (All of them, of course!) And how many scenarios go through a leaf node? Do you see that the leaf nodes match one-to-one with the scenarios? The tree has different "depths" for some scenarios. This is important to keep in mind. For example, some scenarios may describe events where the business is bankrupt.

Node probabilities can be generated in two basic ways. One way is to specify the probability weights of the scenarios, p_s. The probability of a node n is the sum of the probabilities of the scenarios passing through the node

$$p_n := \sum_{s \in \mathcal{S}(n)} p_s$$

The other way is to specify the conditional probabilities of the child nodes as seen from a parent node

$$\Pr\left[\, m \,\middle|\, n \,\right] := \textit{probability of child } m \in n^+$$

The probability of a scenario is the product of the conditional probabilities of the nodes in the scenario

$$p_s = \prod_{n \in \mathcal{N}(s), n > 0} \Pr\left[\, n \,\middle|\, n^- \,\right]$$

6.4.1 Non-anticipativity, Implementability, and Nodes

In our example, the scenario tree models the uncertainty of the demand process. We sometimes say that the demand *generates* the scenario tree.

Suppose now that two scenarios s and σ have a node n in common. As noted above, the tree property means that both s and σ belong to $\mathcal{S}(n)$. For the demand process, specifically, this means that the demand scenarios are *indistinguishable* for all stages $t \leq \tau(n)$

$$d_t^s = d_t^\sigma \quad t = 1, \ldots, \tau(n)$$

The decision variables of our example cannot anticipate the future, so specifically this means that they must also be indistinguishable along those scenarios up to time $\tau(n)$:

$$\begin{aligned} x_t^s &= x_t^\sigma \quad t = 1, \ldots, \tau(n) \\ y_t^s &= y_t^\sigma \quad t = 1, \ldots, \tau(n) \\ z_t^s &= z_t^\sigma \quad t = 1, \ldots, \tau(n) \\ I_t^s &= I_t^\sigma \quad t = 1, \ldots, \tau(n) \end{aligned}$$

This indistinguishable property is called *implementability* or *non-anticipativity* with respect to the scenario tree generated by the uncertain demand.

To add uncertainty to the problem, we could just add scenario notation to everything and then require implementability:

$$\begin{array}{lll} I_t^s - I_{t-1}^s & = x_t^s - y_t^s & s \in \mathcal{S};\, t \geq 1 \\ x_t^s & \in \left[0, K_{t-1}^s\right] & s \in \mathcal{S};\, t \geq 1 \\ y_t^s & \in \left[0, d_t^s\right] & s \in \mathcal{S};\, t \geq 1 \\ I_t^s & \geq 0 & s \in \mathcal{S};\, t \geq 0 \\ \sum_{t=1}^{T_Z} z_t^s & \leq z_0 & \\ z_t^s & \in \{0, 1\} & t = 0, \ldots, T_Z \\ K_t^s - K_0 & = z_t^s K_Z & t = 1, \ldots, T_Z \\ (x, y, z, I) & \text{implementable} & \end{array}$$

Can you see that the most important condition, implementable, is buried in the very last line? You could ask: implementable with respect to what?

Implementability refers to the node partitions $\mathcal{N} := \{\mathcal{N}_t, t \geq 0\}$. When we express the production-inventory problem in terms of nodes, we get implementability for free:

$$\begin{array}{lll} I_n - I_{n^-} & = x_n - y_n & n \in \mathcal{N}_t,\ t \geq 1 \\ x_n & \in [0, K_{n^-}] & n \in \mathcal{N}_t,\ t \geq 1 \\ y_n & \in [0, d_n] & n \in \mathcal{N}_t,\ t \geq 1 \\ I_n & \geq 0 & n \in \mathcal{N}_t,\ t \geq 0 \\ z_n + \sum_{m \in \mathcal{A}(n)} z_m & \leq z_0 & n \in \mathcal{N}_t,\ t = 1, \ldots, T_Z \\ z_n & \in \{0, 1\} & n \in \mathcal{N}_t,\ t = 0, \ldots, T_Z \\ K_n - K_0 & = z_n K_Z & n \in \mathcal{N}_t,\ t \geq 1 \end{array}$$

Whether one is coding software to solve these problems or writing tools to analyze the results and the quality of the modeling, it is the node structures that represent the dynamics and the uncertainty. The scenario tree *is* the node structure. The nodes are the main point!

> The nodes describe the dynamic "information process" of the stochastic program. When you learn something new at a node, that fact is represented precisely by branching to the child nodes.

Let us study the two terms *implementability* and *non-anticipativity* a bit more. A decision is implementable, i.e., can be implemented, if it does not depend on values that are not yet revealed. "Buy IBM stock now if their values go up next month" is *not* implementable, while "Buy IBM stock now" is. In the same way, "Buy IBM stock now if their values go up next month" is anticipative, i.e., anticipates (uses) information that is not yet known. Non-anticipativity then means that a decision does not use such unavailable information.

The term non-anticipativity is most used. But be a bit careful about how you understand it. The whole point of stochastic programming is to be anticipative, that is, to look into the future and consider what *might* happen. However, you must not anticipate what *will* happen.

6.5 How to Talk About Scenario Trees

In many approaches to multistage decision-making, the scenario tree is understood to be an approximation to a stochastic process that is connected in some way to the data. Chapter 4 in Pflug and Pichler's text [69] has an excellent treatment of various techniques for approximating scenario trees from data generated by stochastic processes. But this poses a huge obstacle in practice. There are two basic reasons:

1. Decision-makers do not use or understand stochastic processes.
2. Stochastic processes are calibrated from past data, but decisions are about the future.

Since this is a book about modeling, we think you should pay attention to the decision-makers. What is the best way for decision-makers to interact with multistage models of uncertainty?

We think the best approach is to describe future uncertainty in terms of *forecasts* and *surprises*. This seems natural to the way humans think about the world. As you will see, however, there are significant technical requirements to describe future uncertainty in this way. First we create a silly example to motivate the idea, and then we develop the technical foundations.

6.5.1 What to Tell the Boss

Suppose you are the forecasting expert for a small warehouse. You have analyzed the data, and you know that on any given day there could be no demand, or there could be demand for as many as forty units. So "usually" the demand is in the range {0, 10, 20, 30, 40}. Without any more information, we may as well assume the demand is uniformly distributed.

Now suppose a customer places 10 orders for delivery the next day. We have the usual situation plus this extra knowledge. How may we represent this knowledge? Clearly, the two distributions are

$$\begin{aligned} &U\{0, 10, \ldots, 40\} \quad \text{for next-day orders} = 0 \\ &U\{10, 20, \ldots, 50\} \text{ for next-day orders} = 10 \end{aligned}$$

When the boss comes in the next morning and asks: "What's the forecast for today?," and what do you say?

Will you say

> *Well boss, the distribution is usually* $U\{0, 10, \ldots, 40\}$*, but we have an order booked so today it will be* $U\{10, 20, \ldots, 50\}$.

What do you think about this answer?

Let us try to think of reactions the boss might have:

- "Why can't we have 52 orders? Just the other week, we had 57!"
- "What on earth does Uniform mean? Are you asking for more uniforms? We have plenty and anyway you never wear a uniform!"

To be honest, while bosses are not always right, we think there is nothing wrong with these reactions. You are driving the boss crazy!

Let us try another answer:

> *On average we get around 20, but there are 10 next-day orders so I wouldn't be too surprised if it is between 20 and 40.*

This is a good answer! It applies an easy-to-understand concept: There is a usual sort of variability, and there is an event that changed the usual sort of variability. It hides the inner workings of your modeling, yes, but the boss hired *you* to worry about inner workings of models.

Your answer has framed the situation into a structure like this:

$$\begin{aligned} &20 + U\{-20, -10, 0, 10, 20\} \text{ when next-day orders} = 0 \\ &30 + U\{-20, -10, 0, 10, 20\} \text{ when next-day orders} = 10 \end{aligned}$$

The first term is a forecast, and the second term is a probability distribution, which we may call the "surprise."

It turns out that this "forecast and surprise" structure has some very interesting properties that are universal to all (well-behaved) stochastic processes, even continuous time processes.

6.5.2 [Technical] Conditional Expectation on Scenario Trees

The main concept we need for the forecast and surprise structure is *conditional expectation*. In general, this is a complicated concept that involves a lot of mathematical machinery. Fortunately, the scenario tree notation and attributes developed in this section are very helpful in understanding this important concept.

To get started, suppose $\{\mathbf{Z}_t\}$ is a sequence of random variables on a scenario tree $\mathcal{S}$. It can also be a vector-valued sequence. The conditional expectation of the future values of $\mathbf{Z}_{t'}$ given events (the nodes) in a stage $t < t'$ is

$$\boldsymbol{E}\big[\,\mathbf{Z}_{t'} \,\big|\, \mathcal{N}_t\,\big]$$

You can see that the conditional expectation notation has two parts. The first part is the thing being integrated, the *integrand*. The integrand defines the sample space for the integration. The second part is the domain (or a collection of domains) of integration, the *condition*. The condition defines the collection of subsets of the sample space over which the integration will be computed.

The sample space for $\mathbf{Z}_{t'}$ is the space of scenarios $\mathcal{S}$. The collection $\mathcal{N}_t$ represents the set of scenarios $\mathcal{S}(n)$ going through a node $n \in \mathcal{N}_t$. For each node, we have

$$\boldsymbol{E}\big[\,\mathbf{Z}_{t'} \,\big|\, \mathcal{S}(n)\,\big] = \frac{\sum\limits_{s\in\mathcal{S}(n)} Z^s_{t'} p_s}{\sum\limits_{s\in\mathcal{S}(n)} p_s} \tag{6.7}$$

It is the expected value over $\mathcal{S}(n)$ divided by the probability of $\mathcal{S}(n)$. But is (6.7) well defined? In scenario trees, the requirement is that the integration does not violate non-anticipativity. By now, you are able to prove that conditional expectation satisfies non-anticipativity if and only if $t' \geq t$.

To be sure we understand this conditional expectation, let us check the extreme cases. It will help a lot if you can work through both of these examples! First, check $t = 0$. Recall that $\mathcal{N}_0$ contains only one node, the root node. And of course, all scenarios pass through the root node! Then it follows that

$$\boldsymbol{E}\big[\,\mathbf{Z}_{t'} \,\big|\, \mathcal{N}_0\,\big] = \boldsymbol{E}\big[\,\mathbf{Z}_{t'}\,\big]$$

Next check $t' = t$. For every $n \in \mathcal{N}_t$, we have

$$\begin{aligned} \boldsymbol{E}\big[\,\mathbf{Z}_t \,\big|\, \mathcal{S}(n)\,\big] &= \frac{\sum_{s\in\mathcal{S}(n)} Z_t^s\, p_s}{\sum_{s\in\mathcal{S}(n)} p_s} \\ &= Z_n \end{aligned}$$

This example is the beginning of an important step in our understanding. We just showed (well, you just showed) that $\boldsymbol{E}\big[\,\mathbf{Z}_t \,\big|\, \mathcal{N}_t\,\big]$ has the value Z_n for every node $n \in \mathcal{N}_t$. But we also know that the way scenario trees work is that

$$Z_t^s = Z_n, \quad n \in N_t$$

so we (you) actually proved

$$\boldsymbol{E}\big[\,\mathbf{Z}_t \,\big|\, \mathcal{N}_t\,\big] = \mathbf{Z}_t$$

This means that a conditional expectation is a random variable! Every random variable has a sample space. Can you see that the sample space of $\boldsymbol{E}\big[\,\mathbf{Z}_{t'} \,\big|\, \mathcal{N}_t\,\big]$ is the nodes $n \in \mathcal{N}_t$?

The consequence of this point of view is a very important property of conditional expectations called the *tower property*.

The Tower Property of Conditional Expectations

Suppose $t_0 < t_1$ and $\mathbf{Z}$ is defined on a scenario tree $\mathcal{S}$. Then

$$\boldsymbol{E}\big[\,\mathbf{Z}_{t_1} \,\big|\, \mathcal{N}_{t_0}\,\big] = \boldsymbol{E}\big[\,\boldsymbol{E}\big[\,\mathbf{Z}_{t_1} \,\big|\, \mathcal{N}_t\,\big] \,\big|\, \mathcal{N}_{t_0}\,\big] \tag{6.8}$$

for every $t \in [t_0, t_1]$.

This says that you can always sneak in an additional conditional expectation over $\mathcal{N}_t$ so long as $t \in [t_0, t_1]$.

Let us prove this using the tools we have developed in this section. We will start with the right-hand side. Notice that there are two conditional expectations, one inside the other. The integrand for the inner conditional expectation is $\mathbf{Z}_{t_1}$, so the sample space is the scenario tree. Apply the definition (6.7). For any node $n \in \mathcal{N}_t$, we have

$$\boldsymbol{E}\big[\,\mathbf{Z}_{t_1} \,\big|\, \mathcal{S}(n)\,\big] = \sum_{s\in\mathcal{S}(n)} Z_{t_1}^s\, p_s / p_n$$

The integrand for the outer conditional expectation is the inner conditional expectation $\boldsymbol{E}\big[\,\mathbf{Z}_{t_1} \,\big|\, \mathcal{N}_t\,\big]$. Can you see that the sample space for the inner conditional expectation is $\mathcal{N}_t$?

The outer condition $\mathcal{N}_{t_0}$ will pick out the subset of nodes in $\mathcal{N}_t$ that pass through the nodes $n_0 \in \mathcal{N}_{t_0}$. The nodes we want are the stage-t descendants $\mathcal{D}_t(n_0)$. Now apply the conditional expectation definition

$$\boldsymbol{E}\big[\,\boldsymbol{E}\big[\,\mathbf{Z}_{t_1}\,\big|\,\mathcal{N}_t\,\big]\,\big|\,\mathcal{D}_t(n_0)\,\big] = \frac{\sum_{n\in\mathcal{D}_t(n_0)}\boldsymbol{E}\big[\,\mathbf{Z}_{t_1}\,\big|\,\mathcal{N}_t\,\big]p_n}{\sum_{n\in\mathcal{D}(n_0,t)}p_n}$$

Notice that $\sum_{n\in\mathcal{D}(n_0,t)} p_n = p_{n_0}$, and expand the inner conditional expectation. Then

$$\begin{aligned}\boldsymbol{E}\big[\,\boldsymbol{E}\big[\,\mathbf{Z}_{t_1}\,\big|\,\mathcal{N}_t\,\big]\,\big|\,\mathcal{D}(n_0,t)\,\big] &= \sum_{n\in\mathcal{D}(n_0,t)}\Big\{\sum_{s\in\mathcal{S}(n)} Z^s_{t_1}p_s/p_n\Big\}\,p_n/p_{n_0}\\ &= \sum_{s\in\mathcal{S}(n_0)} Z^s_{t_1}p_s/p_{n_0}\\ &= \boldsymbol{E}\big[\,\mathbf{Z}_{t_1}\,\big|\,\mathcal{N}_{t_0}\,\big]\end{aligned}$$

where the p_n's cancel and the nested summations consolidate into the expression for $\boldsymbol{E}\big[\,\mathbf{Z}_{t_1}\,\big|\,\mathcal{S}(n_0)\,\big]$. This completes the proof.

Finally, let us apply the tower property to a much simpler result where $t_0 = 0$. Then we have

$$\boldsymbol{E}\big[\,\boldsymbol{E}\big[\,\mathbf{Z}_{t_1}\,\big|\,\mathcal{N}_t\,\big]\,\big] = \boldsymbol{E}\big[\,\mathbf{Z}_{t_1}\,\big] \tag{6.9}$$

for all $t_1 \geq t$. This is also called the tower property, even though it is just a corollary of the result above.

6.5.3 Forecast and Surprise

The forecast and surprise structure is a decomposition property for general processes $\mathbf{Z} = \{\mathbf{Z}_t, t \geq 0\}$ on scenario trees. The forecast and surprise decomposition is very simple. For each $t \geq 0$, all we do is add and subtract the conditional expectation:

$$\mathbf{Z}_{t+1} = \boldsymbol{E}\big[\,\mathbf{Z}_{t+1}\,\big|\,\mathcal{N}_t\,\big] + \big(\mathbf{Z}_{t+1} - \boldsymbol{E}\big[\,\mathbf{Z}_{t+1}\,\big|\,\mathcal{N}_t\,\big]\big) \tag{6.10}$$

But we know that the conditional expectation is a random variable on the nodes $n \in \mathcal{N}_t$. So it is in the information state at stage t! It is like a forecast.

The surprise process is

$$\mathbf{W}_{t+1} = \mathbf{Z}_{t+1} - \boldsymbol{E}\big[\,\mathbf{Z}_{t+1}\,\big|\,\mathcal{N}_t\,\big]$$

The surprise $\mathbf{W}_{t+1}$ is a random variable on the nodes $n \in \mathcal{N}_{t+1}$. We could also call it the "forecast error."

The surprise process has two interesting properties. First, the expected value of the surprise is zero:

$$\begin{aligned}\boldsymbol{E}\big[\,\mathbf{W}_{t+1}\,\big] &= \boldsymbol{E}\big[\,\mathbf{Z}_{t+1} - \boldsymbol{E}\big[\,\mathbf{Z}_{t+1}\,\big|\,\mathcal{N}_t\,\big]\,\big]\\ &= \boldsymbol{E}\big[\,\mathbf{Z}_{t+1}\,\big] - \boldsymbol{E}\big[\,\boldsymbol{E}\big[\,\mathbf{Z}_{t+1}\,\big|\,\mathcal{N}_t\,\big]\,\big]\\ &= 0\end{aligned}$$

by the property (6.9).

Second, surprises at different time steps are uncorrelated. Let $\mathbf{W}_t$ and $\mathbf{W}_{t'}$ be surprises and $t' \geq t$. The correlation is

$$\begin{aligned} \boldsymbol{E}\big[\,\mathbf{W}_t\mathbf{W}_{t'}\,\big] &= \boldsymbol{E}\big[\,\boldsymbol{E}\big[\,\mathbf{W}_t\mathbf{W}_{t'}\,\big|\,\mathcal{N}_t\,\big]\,\big] \\ &= \boldsymbol{E}\big[\,\boldsymbol{E}\big[\,\mathbf{W}_{t'}\,\big|\,\mathcal{N}_t\,\big]\mathbf{W}_t\,\big] \\ &= 0 \end{aligned}$$

The first line is the tower property (6.8), and the second line is because $\boldsymbol{E}\big[\,\mathbf{W}_t\,\big|\,\mathcal{N}_t\,\big] = \mathbf{W}_t$, which we (you) proved above. The last line follows from (6.9).

Even when the processes are vector-valued, as can be confirmed by writing out the separate terms of the correlation matrix, the correlations across time steps for the surprise process are 0.

Here is the statement summarizing the properties we have just developed.

Decomposition by Forecasts and Surprises. *Every process* $\{\mathbf{Z}_t, t > 0\}$ *over a scenario tree can be decomposed into a one-step-ahead forecast and a surprise process:*

$$\mathbf{Z}_t = \boldsymbol{E}\big[\,\mathbf{Z}_t\,\big|\,\mathcal{N}_{t-1}\,\big] + \mathbf{W}_t \tag{6.11}$$

The surprise process

$$\mathbf{W}_t = \mathbf{Z}_t - \boldsymbol{E}\big[\,\mathbf{Z}_t\,\big|\,\mathcal{N}_{t-1}\,\big] \tag{6.12}$$

has mean zero and is uncorrelated across time stages.

You might be surprised that the surprises are not correlated between stages. Certainly the scenario tree can model correlations! So how does the forecast-surprise model do it? Let us compute the covariance between $\mathbf{Z}_{t+1}$ and $\mathbf{Z}_t$:

$$\begin{aligned} \mathrm{Cov}\Big(\mathbf{Z}_{t+1},\,\mathbf{Z}_t\Big) &= \mathrm{Cov}\Big(\boldsymbol{E}\big[\,\mathbf{Z}_{t+1}\,\big|\,\mathcal{N}_t\,\big] + \mathbf{W}_{t+1},\,\mathbf{Z}_t\Big) \\ &= \mathrm{Cov}\Big(\boldsymbol{E}\big[\,\mathbf{Z}_{t+1}\,\big|\,\mathcal{N}_t\,\big],\,\mathbf{Z}_t\Big) \end{aligned} \tag{6.13}$$

The first line is the forecast-surprise form, and the second line follows from

$$\mathrm{Cov}\Big(\mathbf{W}_{t+1},\,\boldsymbol{E}\big[\,\mathbf{Z}_t\,\big|\,\mathcal{N}_{t-1}\,\big] + \mathbf{W}_t\Big) = \boldsymbol{E}\big[\,\mathbf{Z}_t\,\big|\,\mathcal{N}_{t-1}\,\big]\boldsymbol{E}\big[\,\mathbf{W}_{t+1}\,\big] + \mathrm{Cov}\Big(\mathbf{W}_{t+1},\,\mathbf{W}_t\Big)$$

which is zero, since $\mathbf{W}_{t+1}$ has mean zero and is uncorrelated with $\mathbf{W}_t$. Let us put this remarkable result in a box:

> Inter-stage correlations arise from the correlation between $\mathbf{Z}_t$ and the conditional one-step-ahead forecast $\boldsymbol{E}\big[\,\mathbf{Z}_{t+1}\,\big|\,\mathbf{Z}_t\,\big]$.

Scenario trees seem to be arbitrary. We can draw any kind of tree, with any allocation of values to nodes and probabilities to arcs. But no matter how arbitrary, every scenario tree has a structure composed of forecasts and surprises. We reveal this structure when we put back something that is missing: the one-step-ahead forecast.

6.5.4 Information State Using Forecast and Surprise Model

Let us now use the forecast-surprise model to complete our development of the information state for the timeline Table 6.1.

6.5.4.1 Demand

The dynamic states $t > 0$ require observations of the random demand $\boldsymbol{d}_t$. These observations are to be generated by a forecast-surprise model. Let us start with the forecast.

For a silly example, let us suppose our production-inventory system produces ice cream. The weather is clearly an important influence on ice cream demand. So, obviously, we will use the weather forecast! Now think about the weather forecast for a minute. Which information state does the weather forecast belong to?

Let us open our weather app and find the weather forecast for tomorrow. When did we observe tomorrow's weather forecast? We observed it just now. The weather forecast for *tomorrow* belongs to the information state for *today*.

Now let us think about the surprise component, the forecast error. This is going to be generated from a simulation model, and this model is going to require some parameters. Which information state do these parameters belong to?

This is a harder question to answer. Just like our discussion of the relationship between observations and decisions, most likely it does not matter since the weather forecasters observe parameters little by little. Our preference is to group parameter updates with the observations, even if the only reason is to organize state indexing in a clear way: The parameters used to generate the stage t surprise belong to stage t.

The general conclusion of this discussion is that the information state for stage t will require a recipe for the forecast:

> *Recipe for the stage $t+1$ forecast, as required by the final state of stage t.*

and a recipe for the surprise:

> *Recipe for the parameters of the stage t surprise, as required by the demand observation in stage t.*

In general, these elements of the forecast-surprise model have to be included in an updated version of the timeline. We do this in Table 6.2. It makes clear to which stage the forecast and surprise information belongs.

Table 6.2 Updated timeline for the production-inventory-option problem

Stage	$t = 0$	$t = 1$	$t = 2$	$\cdots$
Old State	–	(I_0, K_0)	(I_1, K_1)	$\cdots$
Forecast	–	$\boldsymbol{E}[\boldsymbol{d}_1]$	$\boldsymbol{E}[\boldsymbol{d}_2 \vert d_1]$	$\cdots$
Observation	(I_0, K_0)	d_1	d_2	$\cdots$
Surprise	–	$d_1 - \boldsymbol{E}[\boldsymbol{d}_1]$	$d_2 - \boldsymbol{E}[\boldsymbol{d}_2 \vert d_1]$	$\cdots$
Parameters	K_Z, g_0	(c_1, r_1, h_1)	(c_2, r_2, h_2)	$\cdots$
Decision	z_0	(x_1, y_1, z_1)	(x_2, y_2, z_2)	$\cdots$
Rewards	$-C_0$	$R_1 - C_1$	$R_2 - C_2$	$\cdots$
New State	(I_0, K_0)	(I_1, K_1)	(I_2, K_2)	$\cdots$
Next Forecast	$\boldsymbol{E}[\boldsymbol{d}_1]$	$\boldsymbol{E}[\boldsymbol{d}_2 \vert d_1]$	$\boldsymbol{E}[\boldsymbol{d}_3 \vert d_2]$	$\cdots$

6.5.4.2 Prediction

In our silly example, we said that the demand for ice cream was influenced by the weather, so the recipe for ice-cream forecast and surprise has to include the weather. It might include a lot of other influences, too, like traffic or school holidays.

Forecasts are usually the output of a prediction model. Most businesses, and certainly almost all large enterprises, will use prediction models. These prediction models will have lots of inputs (features). The question for us, as stochastic programmers, is whether we need to simulate all those features in our forecast and surprise recipes.

Our general point of view on including all those features is: Oh my goodness, we hope not! We hope that the developers of the prediction model have done a good and thorough job. What do we mean by that? First, a good and thorough model will have predictions for multiple time horizons that correspond to significant business metrics. Second, a good prediction model will give guidance on the forecast errors, or surprises.

Finally, let us turn this around the other way: as Table 6.2 shows, a good forecast and surprise model is a *requirement*.

6.6 Modeling the Horizon

The production-inventory problem, like many inherently multistage problems, in fact has infinitely many stages. By that we do not imply that the problems cover infinitely long time intervals, but that it is not known when the problems end.

A production company knows, of course, that eventually a certain product will go out of production, but they do not know when so they plan as if the production will go on forever. A financial investor will, sooner or later, go out of business, but he plans as if he will not. That is the nature of most decision problems. We really have no choice but to treat these problems as if they had infinitely many stages. But

there is no way we can, technically speaking, handle models with infinitely many stages. We need to make some kind of simplification or approximation to make the model have finitely many stages—if possible, only two.

There is also a modeling related reason for this: As the number of stages gets very large, we approach a kind of steady state situation. But as pointed out before, stochastic programming is about transient behavior, not steady state behavior. We are interested in what to do *now*, not what to do when (if) we eventually reach a steady state. So at best we need to represent the steady state in the model to obtain the right transient behavior. But what actually to *do* once we get there is not at all our focus.

6.6.1 Discounting

In problems with many time stages stretching into the far future, we may want to account for the fact that a payment in the future is less valuable than a payment today. This may be a natural way to create a horizon: At some point the discount is so great that the future does not matter.

For example, we could take a dollar today and invest it in a secure deposit account that pays interest, say, r during each period t. One dollar invested today in this account for t periods would hold

$$\prod_{\tau=1}^{t} (1+r)$$

The inverse of this value represents the amount we would put into the deposit account today in order to get exactly one dollar at time t. This ratio $\delta = (1+r)^{-1}$ is called a *discount rate*. As time increases, discounting progressively lowers the value of a future dollar. For instance, at 15 interest rate, today's value of a dollar to be received 5 years from now is \$0.50 and 10 years from now is \$0.25.

The *discounted present value* of a future cash stream $f = [f_1, \ldots, f_T]$ is

$$\sum_{t=1}^{T} \delta^t f_t$$

This formula simply adds up the various amounts we would have to put into our deposit account in order to generate each term of the payment stream f.

As time goes on, the discounting reduces the impact of the future on present decision-making and imposes a kind of "significance" horizon on comparisons between future cash streams. Discounting is a very common practice in planning problems, but we do not intend you to always accept such smoothing when it appears. For example, is it sensible to discount global warming or nuclear conflict? It does not take much of a discount factor to smooth away the impact of serious events that are far away in time.

6.6.2 Dual Equilibrium

Suppose in the production-inventory problem that there is a time point T in the future when we will not be too concerned about randomness. Can we actually collapse the entire future into a single stage? In other words, create a new stage T and just end the problem right there?

Dual equilibrium, invented by Grinold [29], compresses the long discount tail into a fixed horizon stage T. The idea is to approximate the beyond-the-horizon shadow prices for violation of the equations or inequalities. Of course this means that we are choosing to overlook feasibility violations in order to set the horizon! But as we pointed out initially in this book, hard constraints are not our favorites in any case.

This may not be the correct approach in every modeling situation. But let us explore what it means in the production-inventory problem. The key features of the problem are:

- Objective to be maximized is sales minus expenses of production and inventory storage.
- Sales, production, and inventory are logically linked by the inventory update equation.
- Production is limited by capacity.
- Inventory cannot go negative.
- Sales are limited by demand and implicitly by available production and stock in inventory.

Now let us look for sensible opportunities to apply the tools of the dual equilibrium method: discounting and constraint relaxation. By "sensible" we mean that we can explain it to ourselves and to the business!

Let us suppose the desired horizon is T. We will analyze the "infinity" of stages that follow each horizon node $n \in \mathcal{N}_T$, in the scenario tree $\mathcal{S}$.

The first natural step is discounting. At the horizon node $n \in \mathcal{N}_T$, select a discount factor ρ_n and set the discount rate $\delta_n = 1/(1 + \rho_n)$ as discussed above. We suppose that the discount can depend on the scenario node n, but after that it does not change. We also suppose that the cost and revenue parameters (c_n, f_n, r_n) depend on the terminal node, but do not change after that, and that demand is the expected value conditional on n.

Next, identify the data, and solutions for $t \geq T$ following the terminal nodes $n \in \mathcal{N}_T$:

Expected demand	$d_{n,t} = E[\boldsymbol{d}_t \mid n]$
Production	$x_{n,t}$,
Sales	$y_{n,t}$,
Inventory	$I_{n,t}$, $I_{n,T-1} = I_{n^-}$

where we have assumed that the horizon is beyond the option exercise decision. The notation $x_{n,t}$, et cetera, indicates that value of the decision variable for stages t beyond the horizon T can depend on the node $n \in \mathcal{N}_T$. This is because the continuation value will depend on the leaf node.

The next step of the dual equilibrium method is to evaluate the continuation of the business. At the horizon node $n \in \mathcal{N}_T$, the business has three assets: production capacity K_{n^-}, the inventory stock $I_{n,T-1}$, and the estimate of expected future demand $d_{n,t}$, $t \geq T$. The evaluation of the business is done by modeling the discounted future cash flows.

Our first step is to simplify the production-inventory model by creating an "excess demand" variable $g_{n,t}$ for the production-inventory system and replacing the sales variable by

$$y_{n,t} = d_{n,t} - g_{n,t}$$

The inventory update equation from (6.2) becomes

$$g_{n,t} = d_{n,t} - \left[x_{n,t} + \left(I_{n,t-1} - I_{n,t}\right)\right] \tag{6.14}$$

We grouped this equation into two terms. The first is demand, and the second is the available product from production and from inventory drawdown. So another description of g is the difference between demand and production. When the business produces more than is demanded (g is negative), the surplus production goes into inventory. So it is always the case that $g \geq 0$.

Next let us suppose that there are market prices $u_{n,t}, t \geq T$ at which the business can always buy enough to satisfy demand with equality. The terms in the profit equation are as follows. First, of course, the business must pay the costs of production and inventory storage. The revenue comes from sales, at prices r_n, for the demand $d_{n,t}$ that arrives at the business. The last item is the cost of purchasing product from the market when demand is greater than production. The profit expression is

$$r_n d_{n,t} - c_n x_{n,t} - f_n I_{n,t-1} - u_{n,t} g_{n,t}$$

Now incorporate this into a horizon valuation model at node $n \in \mathcal{N}_T$:

$$\max_{x,I} \quad \sum_{t \geq T} \delta_n^{t-T} \left(r_n d_{n,t} - c_n x_{n,t} - f_n I_{n,t-1} - u_{n,t} g_{n,t}\right)$$

$$\text{such that} \quad \begin{cases} g_{n,t} = d_{n,t} - x_{n,t} - I_{n,t-1} + I_{n,t} & t \geq T \\ x_{n,t} \in [\,0,\, K_{n^-}\,] & t \geq T \\ I_{n,t} \geq 0 & t \geq T \\ g_{n,t} \geq 0 & t \geq T \end{cases}$$

Next, the dual equilibrium approach takes the point of view that the prices are *dual variables*: They take the role of dual multipliers for the constraint $g_{n,t} \geq 0$. A further simplification is to assume that the market prices, or dual variables, are the same for all future periods: $u_{n,t} = u_n$ for all $t \geq T$.

Substituting for g in the valuation problem gives the horizon "Lagrangian"

$$L(u, x, I) = \sum_{t \geq T} \delta_n^{t-T} \Big(r_n d_{n,t} - c_n x_{n,t} - f_n I_{n,t-1} - u_n \left(d_{n,t} - x_{n,t} - I_{n,t-1} + I_{n,t} \right) \Big)$$

Let us now simplify the Lagrangian. First we introduce the horizon variables right-hand sides and variables:

$$\begin{aligned} d_n^* &:= \textstyle\sum_{t \geq T} \delta_n^{t-T} d_{n,t} \\ x_n^* &:= \textstyle\sum_{t \geq T} \delta_n^{t-T} x_{n,t} \\ I_n^* &:= \textstyle\sum_{t \geq T} \delta_n^{t-T} I_{n,t} \end{aligned}$$

Next, there is a tricky calculation for the infinite sums of the $I_{n,t-1}$ variables

$$\begin{aligned} \textstyle\sum_{t \geq T} \delta_n^{t-T} I_{n,t-1} &= I_{n,t-1} + \textstyle\sum_{t \geq T+1} \delta_n^{t-T} I_{n,t-1} \\ &= I_{n,t-1} + \delta \textstyle\sum_{t \geq T+1} \delta_n^{t-1-T} I_{n,t-1} \\ &= I_{n,t-1} + \delta \textstyle\sum_{t \geq T} \delta_n^{t-T} I_{n,t} \\ &= I_{n,t-1} + \delta I_n^* \end{aligned}$$

Finally, we simplify

$$\begin{aligned} L(u, x, I) &= \textstyle\sum_{t \geq T} \delta_n^{t-T} \Big(r_n d_{n,t} - c_n x_{n,t} - f_n I_{n,t-1} \\ &\qquad - u_n \left(d_{n,t} - x_{n,t} - I_{n,t-1} + I_{n,t} \right) \Big) \\ &= r_n d_n^* - c_n x_n^* - f_n \left(I_{n^-} + (1-\delta) I_n^* \right) \\ &\qquad - u_n \left(d_n^* - x_n^* - I_{n^-} + (1-\delta) I_n^* \right) \end{aligned}$$

The last step of the dual equilibrium method minimizes out the dual variable u in order to form the horizon objective. The only term involving u is

$$- u_n \left(d_n^* - x_n^* - I_{n^-} + (1-\delta) I_n^* \right)$$

By our assumption, the u_n are clearing prices for excess demand. As such, the prices must be non-negative. Now we apply complementary slackness to this term. If the right-hand term is strictly positive, then the minimization in $u \geq 0$ is unbounded below. It follows that the effect of the dual variables is to require

$$d_n^* - x_n^* - I_{n^-} + (1-\delta) I_n^* \leq 0$$

The dual equilibrium horizon problem is therefore

$$\max_{x^*, I^*} \quad r_n d_n^* - c_n x_n^* - f_n \left(I_{n^-} + (1-\delta) I_n^* \right)$$

$$\text{such that} \quad \begin{cases} (1-\delta) I_n^* - I_{n^-} \leq x_n^* - d_n^* \\ x_n^* \in [\, 0, K_{n^-} \,] \\ I_n^* \geq 0 \end{cases} \tag{6.15}$$

Dual equilibrium is a general approach to the modeling of horizon constraints. It can be applied to many types of problems. However, like any general approach, there are assumptions made and the solution must be checked to understand if the assumptions are reasonable.

6.7 Formulation of the Full Capacity Option Model

Now we have all the elements to state a formulation of the stochastic multistage capacity option problem:

$$\max_{x,y,z,I} \quad \begin{array}{l} \sum_{t=1}^{T-1} \sum_{n \in \mathcal{N}_t} p_n \left[r_n y_n - c_n x_n - f_n I_{n^-} \right] \\ \quad + \sum_{n \in \mathcal{N}_T} p_n \left[r_n d_n^* - c_n x_n - f_n \left(I_{n^-} + (1-\delta) I_n \right) \right] \\ \quad - g_0 z_0 - \sum_{t=1}^{T_Z} p_n h_n z_n \end{array}$$

$$\text{such that} \quad \begin{cases} I_n - I_{n^-} = x_n - y_n & n \in \mathcal{N}_t,\ t \in [0, T-1] \\ y_n \in [0, d_n] & n \in \mathcal{N}_t,\ t \in [0, T-1] \\ x_n \in [0, K_{n^-}] & n \in \mathcal{N}_t,\ t \in [0, T] \\ I_n \geq 0 & n \in \mathcal{N}_t,\ t \in [0, T] \\ (1-\delta) I_n - I_{n^-} \leq x_n - d_n^* & n \in \mathcal{N}_T \\ z_n + \sum_{m \in \mathcal{A}(n)} \leq z_0 & n \in \mathcal{N}_t,\ t = 1, \ldots, T_Z \\ z_n \in \{0, 1\} & n \in \mathcal{N}_t,\ t = 0, \ldots, T_Z \\ K_n - K_0 = z_n K_Z & n \in \mathcal{N}_t,\ t \in [0, T] \end{cases} \tag{6.16}$$

All the parts are there: the scenario tree, the production-inventory operations, the purchase and exercise of the option, and the horizon equations.

This formulation supposes that the expected value of profit is a sufficient risk model. This is quite normal for business decision-making, as we discussed in Chap. 3.

Inherently multistage problems have perhaps a wider exposure to bad outcomes than an inherently two-stage model. Each stage will have uncertain gains and losses, of course, and there is a non-zero probability to scenarios with maximum losses at every stage. Furthermore, many industries are affected by business cycles and the longer the horizon the greater the chance of very bad losses. So for long-term investments like this capacity option model, it is a good practice to use a risk

objective composed of an expected value of profit plus the conditional value at risk for profits below a certain loss threshold [91].

An additional concern in multistage modeling is the consideration of time consistency. This is a highly technical research topic. The basic idea is that risk objectives applied to intermediate stages can lead to inconsistencies in risk incentives, see [15] and [70]. The recommendation for multistage modeling is that risk metrics be applied only at the terminal stage.

6.8 [Technical] Decomposition by Inventory Subproblem

The stochastic programming formulation of the capacity option model (6.16) is a (very) large stochastic program. In this model, as in most inherently multistage problems, there are multiple "levels" of dynamics. The information state is the framework for analyzing these levels.

The information state analysis in Table 6.2 shows that there are two levels of dynamics: inventory and capacity. There may be multiple inventory stages for each capacity stage. This suggests a kind of decomposition in which a capacity stage "contains" an inventory subproblem.

6.8.1 Inventory Subproblem

Consider the inventory subproblem for a non-horizon node n:

$$\max_{x,y,I} \quad R_n$$

$$\text{such that} \quad \begin{cases} I_n - I_{n^-} = x_n - y_n \\ R_n = r_n y_n - c_n x_n - f_n I_{n^-} \\ x_n \in [0, K_{n^-}] \\ y_n \in [0, d_n] \\ I_n \geq 0 \end{cases}$$

Table 6.3 lists the inputs and outputs for the subproblem. The only inputs not already contained in the inventory information state are the new demand and the existing capacity, and the only updates listed are to the inventory and reward. Can you explain why?

Suppose it is possible to develop a model that "predicts" the output and update given the input for the inventory subproblem:

$$(I_n, R_n) = \mathcal{R}_n(I_{n^-}, K_{n^-})$$

Table 6.3 Information state for inventory subproblem

	Input	Observe	Update	Output
Inventory	I_{n-}	..	I_n	..
Capacity	K_{n-}	..	..	..
Demand	..	d_n	..	..
Reward	..	..	..	R_n

where the node subscript n refers to the demand observation. At the moment let us not worry about how to create such a predictor. For now consider the implications for the first draft model:

- By predicting inventory state update I_n, the production-inventory equations and constraints can be eliminated.
- By predicting net revenue (or reward) R_n, there is no need to optimize over the variables (x, y, I).

But what are we saying here? We are saying that there is a natural decomposition of the problem by inventory state. If the predictor is reasonably accurate, then we will not need to model the production and sales decisions. All necessary information will be contained in the predictor.

6.8.1.1 Horizon Stage

The information state for the horizon subproblem at a horizon node $n \in \mathcal{N}_T$ is very similar to Table 6.3, except that the demand "observation" is the discounted expected demand d_n^* and there is an additional parameter δ, the discount rate. Again, let us suppose there is a model that predicts the horizon revenue

$$R_n^* = \mathcal{R}_n^*(\delta, I_{n-}, K_{n-})$$

And again, this shows that there is a natural decomposition of the problem by inventory at the pre-horizon stage.

6.8.2 Capacity Option Decision Reformulation

Capacity changes only when a decision to exercise the option is made. In our modeling we supposed that there is a date T_Z after which the option cannot be exercised. Furthermore, a bespoke option like the one we are modeling here is very likely to restrict exercise decisions to particular dates—perhaps quarterly or yearly. The capacity state will change only on those dates.

This leads us to consider a decomposition formulation that highlights the option purchase and exercise decisions. The outline of this formulation is as follows:

1. Revise the stage structure to correspond to the capacity exercise dates.
2. Set the horizon stage T to be one stage past the final exercise date, so $T = T_Z + 1$.
3. Replace the dynamics with the prediction models $(\mathcal{R}, \mathcal{R}^*)$, and attach node subscripts to identify the information state used by the prediction models.

This leads to a reformulation that focuses on the capacity option decisions.

$$\max_{z \in \{0,1\}} \sum_{t=1}^{T-1} \sum_{n \in \mathcal{N}_t} p_n \left[R_n - h_n z_n \right] + \sum_{n \in \mathcal{N}_T} p_n R_n^* - g_0 z_0$$

$$\text{such that} \quad \begin{cases} (R_n, I_n) = \mathcal{R}_n(I_{n^-}, K_{n^-}) & n \in \mathcal{N}_t,\ t \in [1, T] \\ R_n^* = \mathcal{R}_n^*(\delta, I_{n^-}, K_{n^-}) & n \in \mathcal{N}_T \\ z_n \le z_0 - z_{n^-} & n \in \mathcal{N}_t,\ t \in [0, T-1] \\ K_n = K_0 + z_n K_Z & n \in \mathcal{N}_t,\ t \in [0, T-1] \end{cases} \tag{6.17}$$

This model shows that the relationship between production capacity and revenue drives the capacity option decisions.

6.8.2.1 Three Stages

Now suppose that there is only one exercise date, at T_Z. Then there are three capacity stages:

1. First stage sets initial capacity and inventory and decides option purchase.
2. Second stage runs demand scenarios out to exercise date, records revenue from operations, and decides option exercise.
3. Third stage runs demand scenarios over the horizon, decides optimal ending inventory, and records discounted future revenue.

This will require two "prediction" models, one to predict the revenue and ending inventory during the second stage, and one to predict the discounted revenue at the horizon. The first is the "pre-exercise" model and the second is the "post-exercise" model.

6.8.2.2 Pre-exercise Prediction Model

The pre-exercise model will surely include multiple production stages. Let us suppose that the production stages are a selection of dates following the date corresponding to capacity stage $t = 0$ (the option purchase decision) and ending at the exercise date corresponding to stage $t = T_Z$.

The prediction model during the second capacity stage is derived from the first four equations and the first objective term of the full model (6.16).

$$\max_{x,y,I} \quad \sum_{t=1}^{T_Z} \sum_{n \in \mathcal{N}_t} p_n \left[r_n y_n - c_n x_n - f_n I_{n^-} \right]$$

$$\text{such that} \quad \begin{cases} I_n - I_{n^-} = x_n - y_n & n \in \mathcal{N}_t,\ t \in [1, T_Z] \\ y_n \in [0, d_n] & n \in \mathcal{N}_t,\ t \in [1, T_Z] \\ x_n \in [0, K_0] & n \in \mathcal{N}_t,\ t \in [1, T_Z] \\ I_n \geq 0 & n \in \mathcal{N}_t,\ t \in [1, T_Z] \end{cases} \tag{6.18}$$

The output of this model is the predictor for the inventory and revenue pair at node $n \in \mathcal{N}_{T_Z}$ given the initial inventory and capacity:

$$(I_n, R_n) = \mathcal{R}_n(I_0, K_0) \quad n \in \mathcal{N}_{T_Z}$$

The node subscript $\mathcal{R}_n$ identifies which inventory state is used to generate the prediction. By our observe-then-decide protocol, all the components of the production stages in the information state for node n occur *before* the second-stage exercise decision.

Now, we are being vague about what node $n \in \mathcal{N}_{T_Z}$ really means. Did you notice that the second-stage prediction model (6.18) requires a scenario tree for the demand? And then we said that the capacity option model only needs the output of the prediction model? Does $n \in \mathcal{N}_{T_Z}$ mean demand scenario tree or capacity option scenario tree?

Answering this question is probably the most important step in modeling multistage stochastic programs! Let us bookmark this question for now and look at the third-stage model. The same question will arise there.

6.8.2.3 Post-exercise Prediction Model

The post-exercise model will also likely include multiple production stages, just like the second capacity stage model. Again, let us identify a selection of dates following the exercise date T_Z and ending with the horizon date T. Now derive the third capacity stage model from the full model (6.16).

$$\max_{x,y,I} \quad \begin{array}{l} \sum_{t=T_Z+1}^{T-1} \sum_{n \in \mathcal{N}_t} p_n \left[r_n y_n - c_n x_n - f_n I_{n^-} \right] \\ \quad + \sum_{n \in \mathcal{N}_T} p_n \left[r_n d_n^* - c_n x_n - f_n \left(I_{n^-} + (1-\delta) I_n \right) \right] \end{array}$$

$$\text{such that} \quad \begin{cases} I_n - I_{n^-} = x_n - y_n & n \in \mathcal{N}_t,\ t \in [T_Z + 1, T - 1] \\ y_n \in [0, d_n] & n \in \mathcal{N}_t,\ t \in [T_Z + 1, T - 1] \\ x_n \in [0, K_n] & n \in \mathcal{N}_t,\ t \in [T_Z + 1, T] \\ I_n \geq 0 & n \in \mathcal{N}_t,\ t \in [T_Z + 1, T] \\ (1-\delta) I_n - I_{n^-} \leq x_n - d_n^* & n \in \mathcal{N}_T \end{cases} \tag{6.19}$$

The output of this model is the predictor for the for the option exercise decisions

$$R_m^* = \mathcal{R}_m^*(\delta, I_n, K_n) \quad n \in \mathcal{N}_{T_Z}, m \in \mathcal{D}(n, T)$$

The node subscript m identifies the node in the scenario tree corresponding to the discounted demand at the horizon d_m^*. The third-stage model is used to evaluate the impact of the option exercise decision z_n at nodes $n \in \mathcal{N}_{T_Z}$ on the revenue R_m^* distributed over the descendant nodes $m \in \mathcal{D}(n, T)$.

Again, here is the same question. When we (vaguely) refer to nodes in $\mathcal{N}_{T_Z}$ or $\mathcal{N}_T$, which scenario tree do we mean? Do we mean the scenario tree for demand, or the scenario tree for revenue?

6.9 [Technical] Transient and Steady State Decomposition

Our framing of the capacity option model decomposes it into "transient" and "steady-state" components. The transient ones are the option purchase and exercise decisions. These decisions are made only once. The capacity option purchase model (6.17) is the stochastic program we believe best represents the transient components of the option purchase problem.

In this final section we discuss the remaining modeling issues:

- Predictors for steady state components (6.18) and (6.19).
- Generators for time-series data for training predictors.
- Incorporate decision-maker counterfactuals into the steady state model.

The discussion will necessarily be high level. Full treatments of some of these issues deserve entire chapters, or even books. Moreover, technology (but perhaps not the mathematics) for predictors and generators is evolving rapidly.

6.9.1 Predictors for Steady State Solutions

As noted above, the main goal of the predictor is to estimate the probability distribution of the revenue: $\mathcal{R}$ and $\mathcal{R}^*$.

Stochastic programming algorithms, such as Stochastic Dual Dynamic Programming, iteratively approximate the value function for steady state problems. This is a very effective algorithm and has become a workhorse for multistage stochastic programming. SDDP was proposed by Pereira and Pinto in [67] and is still a very active area of research. Some recent contributions are [59, 92, 16]. However, SDDP suffers from the curse of dimensionality. A recent proposal for a machine learning approach can be found in [13].

Steady state problems are also very good candidates for approximation using machine learning technologies. A natural option is Reinforcement Learning; see the

recent book by Warren Powell [72]. Reinforcement Learning and related algorithms like Monte Carlo Tree Search have the advantage that they are self-supervising.

Predictors for non-linear dynamical systems are also a very active and productive area of research; see the Wikipedia article [88] for references. You may be surprised to learn that it is easier to predict optimal solutions—optimality imposes structure on feature space modeling. For example, predictors have now been trained for very high-dimensional energy minimization problems like protein folding [45]. On the other hand, these are supervised learning algorithms and as such require the preparation of labeled data.

Finally, it is worth mentioning that there are very well-studied approximation algorithms for production-inventory models, like models with lot sizing as we mentioned above. Some of these algorithms develop decision rules, like base stock inventory policies. A recent paper in this literature is [12]. If these policies give reasonable results, there is no reason not to use them for prediction.

6.9.2 Generators for Time Series

The training of predictors, whether for Stochastic Dual Dynamic Programming, Reinforcement Learning, or Physics-Informed Neural Networks, requires simulation of very large numbers of sample scenarios: thousands, millions, or billions. Even if one is using a production-inventory decision rule, the generation of the distribution of revenue will require simulations of sample scenarios to recover the dependence structure in the time series. This topic has been extensively discussed in Chap. 4.

6.9.3 Counterfactuals

The time-series generator used to train the predictors will be more useful to decision-makers if they can incorporate "counterfactuals." This term refers to the generation of simulations with features that are not present in past data.

Counterfactuals may refer to different decisions taken in the past. For example, after a banking crisis, regulators naturally want to know what would have happened if they imposed a different set of rules. For the capacity option model, your boss may want to know what would have happened if they had changed prices in the past.

Another form of counterfactual refers to different parametrization of the demand generator models. For example, bank regulators in the United States require the big banks to evaluate their risks under counterfactual scenarios that model future challenges like an unforeseen recession, or an unexpected burst of inflation. Your boss may want to evaluate the capacity option decisions under counterfactuals that represent changes in technology or competition.

Counterfactuals for our capacity option model require the addition of new random variables to the information state. For example, we can create a "new competitor" counterfactual, but then we need to model the impact of this counterfactual on the revenue. This could be modeled as a random percentage of market share taken by the new competitor, and added to the information state.

Counterfactuals for changes in the time-series generator will also require new variables in the information state. For example, changes in the inter-stage dependency model can be modeled by changes in the regression or copula parameters and represented as a counterfactual in the information state. Changes in the statistics of the intra-stage marginals can be modeled by changes in the moments and quantiles and also represented as a counterfactual in the information state.

Chapter 7
Service Network Design

With Arnt-Gunnar Lium and Teodor Gabriel Crainic

The shortest distance between two points is under construction.
– *Noelie Altito*

This chapter represents an investigation following the lines of this book and where the focus is that of a graduate student studying the effects of uncertainty on a specific problem. There is no customer in this problem, and it has not reached the level of sophistication needed for a real application. However, it goes to the heart of this book: What does stochastics do to my problem? What are the implicit options? This chapter is based on the PhD thesis of Arnt-Gunnar Lium of Molde University College. For an overview see [60]. You are going to meet an inherently two-stage problem with, principally, infinitely many stages. However, since we in this situation do not really need the decisions of the inherent second stage, we can approximate, ending up with a two-stage model. In our view, this points to the heart of stochastic programming: inherently two-stage problems with rather complicated stages after the first one.

7.1 Cost Structure

The starting point here is the problem of less-than-truckload trucking (LTL), that is, trucking services set up to service the market of small packages, where renting a full truck is out of question. The problem facing the planner is to decide how many trucks to buy (or rent) and which schedules to set up for these trucks. In the example we present here, the trucks have weekly schedules. As flows of goods are usually quite unbalanced, the movement of empty trucks for repositioning is crucial to capture. Also, trucks may at times be parked during full days, simply because they are not needed that day. We shall let the daily rate of having a car parked be lower than that of having it run.

A. J. King, S. W. Wallace, *Modeling with Stochastic Programming*, Springer Series in Operations Research and Financial Engineering,
https://doi.org/10.1007/978-3-031-54550-4_7

This cost structure has two different interpretations. We can assume we rent the trucks (long term, not day by day), and the cost of having the truck parked equals the pure renting costs. Then, for each of the movement costs, by subtracting the cost of a parked truck, we find the variable costs of actually running the truck along a certain route. Alternatively (since we work by the week), seven times the daily parked rate can be interpreted as the weekly fixed costs of having a truck, and the differences, just as above, are the variable costs. Hence, the model covers both a pure variable cost interpretation and a fixed and variable cost interpretation. If you only need to pay for trucks actually being used, the problem changes considerably, and that version is not covered in this chapter.

7.2 Warehouses and Consolidation

A package does not necessarily follow a given truck from its origin to its destination. Rather, the whole point of the service network design is to set up truck schedules, so as to facilitate the transportation by using warehouses and consolidation of goods. So packages may spend some of their time in one truck, arrive at a warehouse, maybe stay some time at a warehouse, and then continue with another truck. It can, in principle, spend time on many trucks before arriving. Each package has, in addition to a starting and an ending node, a day of being available in its starting node and the latest day of arrival. Clearly, the longer you are allowed to keep the package, the more "interesting" routes you can find to efficiently use your trucks' capacity, and therefore, the more money you can make. On the other hand, in reality, the quicker you get it to its destination, the higher prices you can charge. In this model a package simply has a date of arrival at its starting node and a due date. But while modeling, the above trade-off must be considered.

7.3 Demand and Rejections

The next major modeling issue here is that of how to model orders and potential rejections. This is, from a modeling perspective, rather complicated. And we would clearly advise you to think carefully about it. You may have a history of demands. Should that be used as a measure of potential future demands? If you answer yes, you may be indicating that even though you are about to create a new (and better) service network (new schedules), you will observe the same demand as before. Is that reasonable? You may, of course, also look at the full potential demand for your services, also including the demand presently being picked up by competitors. But is that reasonable? Probably not.

Let this question of how to model the demand rest a short while, and let us look at another aspect of the model—as they are connected. Will you reject orders that are not profitable or that you have no capacity to carry? Traditionally, deterministic

LTL models operate with expected demands (maybe called estimated demands) or something slightly larger to account for normal variations and at planning level assume that all demand must be met. Or to put it differently, that these demands are what we set up the service network for. They realize, of course, that in real time, orders will be different, and rejections may take place.

Hence, the modeling framework is that of planning to deliver the expected demand (or something slightly larger), but realizing that while running the system, you may in fact reject. When formulating a service network design model with random demand, you may have to rethink this strategy. Is it really correct to assume that you can carry it all? If you set up a two-stage stochastic program, with demand being stochastic and where the first stage is to set schedules (and determine the number of trucks) and the second stage to send the packages, should you then require that demand can be met with your trucks in all scenarios?

Let us see what that would imply. If the demand you use in your model is based on history, for example, by simply using observed demand as scenarios (see Sect. 4.1.2 for a discussion), it should be obvious (right?) that the most extreme (high-demand) scenarios will drive the solution. You may have to hire a full truck to transport one package in one single scenario (and that is all you use the truck for throughout a week and only if that scenario occurs). That is clearly unreasonable in most interpretations of the model.

If you use all potential demand to estimate scenarios, an approach of being able to send anything is obviously unreasonable, so we shall not discuss that any further.

So, it seems the only reasonable model is to allow rejections in the model itself, that is, to plan to reject certain demands. Many will react negatively to explicitly plan to reject orders—because it may, for example, create bad publicity. Even so, the fact is, also in the deterministic modeling framework, we planned to reject in real time. It is just that it did not show in the tactical model.

But what stochastic demand should we use? There is no right answer here. The right answer only exists is you *know* the future, stochastically speaking, and you do not. That is a strong statement, so reflect on it for a while. Except for such problems as playing at a Casino (where the rules are known, and we hope the deck of cards is proper), can we know the future in terms of distributions? In all other cases, we simply hope that we know the future. And we cannot test if our view on the future is good or bad. We can test if the past was a good description of the future in the past. But that is as far as we can get.

> We hardly ever actually know the relevant distributions, as our interest is the future, and we do not know (and cannot test) if the past describes the future.

So far we have discussed two possible distributional assumptions that the future is described by our own past and that we use all potentially available demand to

estimate our scenarios. We hope you see fairly quickly that the latter is not a good idea. It would overestimate the demands available to us.

If this was a real problem you were to solve, finding the correct distributional assumptions would be hard. In fact, it might turn out to be the hardest part of the modeling exercise. We shall now leave that behind and simply assume we have a distribution describing our best estimate of future (random) demand, and we shall assume that packages can be rejected. But we hope this discussion has shown that the issues involved are complicated, both with respect to distributions and whether or not a constraint should be hard or soft (i.e., should be enforced or deviations penalized).

7.4 How We Started Out

The above discussion was far from obvious to us when we started out. First, we started out with demands that represented what we *had* to transport (so no rejection—in line with the deterministic models) and the scenario generation method of Sect. 4.4.2. We were unable to obtain in-sample stability as defined in Sect. 4.2. So we asked why? The reason was, in our view, an interesting effect of two phenomena.

1. The scenario generation method of Sect. 4.4.2 makes trees that match four marginal moments and correlations. It also keeps the approximate shape of the initial marginal distributions put into the iterative procedure of the method. But, and this turned out to be important, the extreme values were not exactly the same in all scenario trees.
2. In an integer program of this type, an arbitrarily small increase in the maximal possible demand can result in the need for an extra truck and hence a jump in costs. Alternatively, it can result in substantially different schedules (but the same number of trucks), also resulting in jumps in the cost.

So, we were concerned about the instability. Note that this concern is not only about lack of numerical stability (and hence the ability to solve) but also a modeling concern. What is it with our model that makes it jump around like this? Remember what we have pointed out before, the scenario tree generation is part of our modeling, as is the algebraic description of the model. A good model, in this extended sense, would be stable.

So, our first step was to extend the method to handle bounded support, i.e., guarantee that no outcomes were generated outside the support. That turned out to be feasible, and we observed in-sample stability. But we were still concerned. A close look at the solution showed that some of the trucks were hardly used at all, and they were only needed to transport packages with very small probabilities of showing up. And we had to ask: Is that how such companies are run? The answer was obviously no. That made us switch to the model with rejections. What rejection cost to choose is of course a question in its own right. If rejection means sending

the packages with a competitor, the rejection cost could simply be the extra cost of using the competitor. If, in addition, there is concern about reputation, it may be set a bit higher. If rejection is literally a rejection, lost customers and reputation must both be included. Rejection may also refer to you sending the packages yourself, but with a mode not covered by the model.

So, out of these stability tests came a revision of both the algebraic model and the modeling of scenarios. It is worth repeating that in-sample stability is not only a numerical issue but also a modeling issue. A model that is not in-sample stable is not a good model.

7.4.1 The Stage Structure

As it stands, this problem is inherently two-stage, with inherent Stage 1 being the setup of the network and inherent Stage 2 the use of the network. The number of actual stages in the problem is large (possibly infinitely large), but we do not want to model that. In fact, having infinitely many stages in a transient model will not do. On the other hand, our interest is the design, not the dispatch. We shall use that to arrive at a two-stage model. It is worth noting what is going on here. The way we shall model commodity flow will represent the flow well enough to get a good design: The model will understand that the design must function under many different demands. But the commodity flows will *not* represent possible dispatches in reality, as they do not capture the dynamics of demand realizations.

> In inherently two-stage problems we are not really interested in the variables belonging to the inherent second stage. Hence, we can simplify as long as the effects on the first stage variables are kept.

However, we cannot remove the variables, as that would remove the signal for how to obtain good robust schedules. Many inherently two-stage problems can be modelled this way—by realizing that the variables of the inherent second stage are not really needed for implementation.

7.5 A Simple Service Network Design Case

The proposed model starts from a deterministic fixed cost, capacitated, multi-commodity network design model. Integer-valued decision variables are used to represent service selection decisions, while product-specific continuous variables capture the commodity flows. For instances where the service offered is not able

to meet the demands, we introduce a penalty cost (or an outsourcing cost). The penalty can refer to plain rejection, the use of a competitor, or possibly sending the commodity with another service offered by the company itself, be that a more expensive or slower service. The goal is to minimize the total system cost under constraints enforcing demand, service, and operation rules and goals.

Several simplifying assumptions are made:

- We consider a homogeneous fleet of capacitated vehicles with no restrictions on how many vehicles are used.
- The transport movements require one period, while terminal operations are instantaneous (within the period).
- Demand cannot be delivered later than the due date but may arrive earlier.
- There is a (fixed) cost associated with operating a vehicle (service), but no cost is associated with moving freight (except when outsourcing is used); that is, truck movements cost the same whether the trucks move loaded or empty.
- There are no costs associated with time delays or terminal operations.
- The plan is repeated periodically.

The space–time network is built by repeating the set of nodes (terminals) $\mathcal{N}$ in each of the periods $t = 1, \ldots, T$. Each arc (i, j) represents either a service, if $i \neq j$, or a holding activity, if $i = j$. A cost c_{ij} is associated with each arc (i, j), equal to the cost of driving a truck from terminal i to j if $i \neq j$ or to the cost of holding a truck at terminal i if $i = j$. The cost of outsourcing one unit or a failed delivery of a unit is represented by b. A period-by-period complete network is assumed. For each commodity $k \in \mathcal{K}$, we define its random demand $\delta(k)$, origin $o(k)$, destination $d(k)$, and the periods $s(k)$ and $\sigma(k)$ when it becomes available at its origin and must be delivered (at the latest) at its destination, respectively. The truck capacity is denoted M.

Stochastics are described in terms of scenarios $s \in \mathcal{S}$. To each scenario is attached a probability $p^s \geq 0$, with $\sum_{s \in S} p^s = 1$. A scenario is $\mathcal{K}$-dimensional, as it contains one demand for each commodity. To indicate the impact of demand variation, the Y and Z flow variables are now indexed by s. The demand for commodity k in scenario s is given by $\delta(k, s)$.

In any multi-period formulation, one has to address end-of-horizon effects. We can mitigate this problem by casting the model in a circular fashion. Assuming a T-period planning horizon, a circular notation means that the period preceding period t is given by

$$t \ominus 1 = \begin{cases} t - 1 & \text{if } t > 1 \\ T & \text{if } t = 1 \end{cases} \tag{7.1}$$

This approach is allowed since this is an inherently two-stage problem where our real interest is the design variables, not the flow variables. So we capture the effects of uncertainty, but the resulting dispatch is useless.

The decision variables and model are:

$Y_{ij}^{t\ominus 1}(k,s)$: amount of commodity k going from terminal i in period $t\ominus 1$ to terminal j in period t in scenarios s
$X_{ij}^{t\ominus 1}$: number of trucks from node i in period $t\ominus 1$ to node j in period t
$Z(k,s)$: amount of commodity k sent to its destination using outsourcing or simply not delivered at all in scenario s

Note that the circularity of the network means that we can have $\sigma(k) < s(k)$; if we have, for example, a commodity in a 1-week network ($T = 7$) with $s(k) = 5$ and $\sigma(k) = 1$, it would mean that it becomes available on Friday (each Friday) and must be delivered by the following Monday. To facilitate this fact in the mathematical formulations, we define a set of periods that a commodity can be shipped from

$$\mathcal{F}_c = \{s(k), \ldots, \sigma(k) - 1\} \text{ if } \sigma(k) > s(k) \text{ and } T \setminus \{\sigma(k), \ldots s(k) - 1\} \text{ otherwise} \tag{7.2}$$

The stochastic problem that we solve is

$$\min \sum_{ij\in\mathcal{N}} \sum_{t=1}^{T} c_{ij} X_{ij}^{t} + b \sum_{s} p^{s} \sum_{k\in K} Z(k,s) \tag{7.3a}$$

$$\text{s.t.} \sum_{i\in\mathcal{N}} X_{ij}^{t\ominus 1} = \sum_{i\in\mathcal{N}} X_{ji}^{t} \qquad j \in \mathcal{N}, t \in T \tag{7.3b}$$

$$\sum_{i\in\mathcal{N}} Y_{ij}^{t\ominus 1}(k,s) - \sum_{i\in\mathcal{N}} Y_{ji}^{t}(k,s) = \begin{cases} \delta(k,s) - Z(k,s) & \text{if } j = d(k) \text{ and } t = \sigma(k) \\ -\delta(k,s) + Z(k,s) & \text{if } j = o(k) \text{ and } t = s(k) \\ 0 & \text{otherwise} \end{cases}$$

$$j \in \mathcal{N}, t \in T,\ k \in K,\ s \in \mathcal{S} \tag{7.3c}$$

$$\sum_{k\in K} Y_{ij}^{t}(k,s) \le M X_{ij}^{t} \qquad ij \in \mathcal{N},\ i \ne j,\ t \in T,\ s \in \mathcal{S} \tag{7.3d}$$

$$0 \le Z(k,s) \le \delta(k,s) \qquad k \in K,\ s \in \mathcal{S} \tag{7.3e}$$

$$X_{ij}^{t} \ge 0 \text{ and integer} \qquad ij \in \mathcal{N},\ t \in T \tag{7.3f}$$

$$0 \le Y_{ij}^{t}(k,s) \le \delta(k,s) \qquad k \in K,\ ij \in \mathcal{N},\ t \in T,\ s \in \mathcal{S} \tag{7.3g}$$

$$Y_{ij}^{t}(k,s) = 0 \text{ for } t \notin \mathcal{F}_k \qquad k \in K,\ ij \in \mathcal{N},\ s \in \mathcal{S}. \tag{7.3h}$$

The objective function (7.3a) minimizes the sum of the first-stage costs for opening the arcs plus the expected rejection costs. The constraints in (7.3b) enforce conservation of flow for trucks; the "$t \ominus 1$" expression in the subscript ensures that the trucks form circular routes. The constraints in (7.3c) play a similar role for the

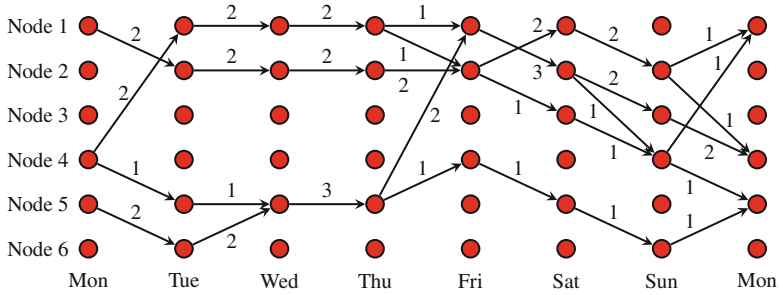


Fig. 7.1 Typical first-stage output from the model. The first and last columns represent the *same* day. Numbers on arcs refer to the number of trucks running in parallel on the arcs

commodity flow, except that the flow has an origin and a destination, instead of being circular. The volume of the flow is equal to the satisfied demand, i.e., the difference between the scenario demand $\delta(k, s)$ and the rejected demand $Z(k, s)$. The constraints in (7.3d) limit the total flow on each arc to the booked volume on the trucks, and the constraints in (7.3e) limit the rejected demand to the actual scenario demand. Note that (7.3d) does not apply on holding arcs with $i = j$, since we assume that commodities can be held at the nodes without a truck being present and with no capacity restriction.

Finally, the constraints in (7.3h) ensure that the path for commodity c does not loop around the network and thus prevent deliveries taking more than T time periods. We could also fix to zero all the Y variables in the origin period $o(k)$ that do not start from the origin node $d(k)$ and do the same for the destination, but it would make the formulation more difficult to read; in addition, this can be expected to be taken care of by an LP presolver.

When this model is solved, we obtain how many trucks to use, their fixed routes, the flow of commodities (one set of flows per scenario), the amounts outsourced (also by scenario), and finally, the total expected cost. Figure 7.1 shows a typical example of truck routes from the model.

It is worth noting the interpretation of the flows of commodities. We get one set of flows per scenario. Hence, when flow demands actually arrive in sequence (and not scenario by scenario), we cannot use the flows directly for routing goods. However, we chose this approach as it helps us achieve good robust schedules, avoids the end-of-horizon effects, and at least provides guidance for flow.

> In an inherently two-stage model, we are not really interested in decisions in later stages, but we need to send good signals to the first stage. This opens up for useful approximations.

7.6 Correlations: Do They Matter?

As an example of how ideas from this book can be used to study a new problem, let us ask ourselves the question: How important are correlations? Does it matter if we get them right or not? We must remember that deterministic models have nothing that corresponds to correlations. So if they are important, we need a tool that can relate to them. We outlined this in Sect. 1.8.2.

To study correlations, a scenario generation procedure allowing us to actually control correlations is needed. We have used a slightly updated version of the method in [43]. That method allows us to control the first four moments of the marginal distributions, plus the correlations. The extension makes sure that we stay within the support of the random variables as we have already explained. It should be clear that had we used sampling to obtain scenarios, we could not have controlled the correlations (nor the marginal distributions for that matter) unless we created very large trees. But with large trees, the model would not be numerically solvable.

With this approach we obtained in- and out-of-sample stability as outlined in Sect. 4.2. Let us see where this takes us. With an in-sample stable scenario tree and an optimization problem of limited size, we can investigate our main concern, the importance of correlations for the optimal solution.

7.6.1 Analyzing the Results

In this example, demand is described by triangular distributions, each defined by its min, mode, and max. In addition, we have a correlation matrix. Due to the complexity of the problem, and for a better overview of the results, we limit ourselves to use no more than 16 random variables and 252 integer variables.

Flexible or Robust?
Let us first reflect on flexibility and robustness. What do we actually want to achieve in this case? The schedules, as seen by the customers should be robust, that is, it should not be necessary to change them in the light of shocks. This aspect is not really part of our model, as we by defining the commodities have required this robustness to be present. In our model there is no possibility of dropping a service. It is feasible to outsource a full service, but from a customer's perspective, the service is still there. The services, as seen by customers, refer to what services are offered, not how the goods are transported. So this aspect is part of the input to the model and is taken care of that way.

Second, by *requiring* the model to have schedules for the trucks, we also immediately achieve robustness for drivers and trucks. That is, the varying demand does not result in repeatedly new plans for the trucks and drivers. We might have defined models where trucks could be rerouted, but we did not. So, the resulting schedules are robust, as required, from the drivers' perspective.

0 Correlation				Mixed Case				0.8 Correlation			
1	0	0	0	1	0.8	-0.8	-0.8	1	0.8	0.8	0.8
0	1	0	0	0.8	1	-0.8	-0.8	0.8	1	0.8	0.8
0	0	1	0	-0.8	-0.8	1	0.8	0.8	0.8	1	0.8
0	0	0	1	-0.8	-0.8	0.8	1	0.8	0.8	0.8	1

Fig. 7.2 Structure of the three correlation matrices used in the tests

Flexibility in Routing

The correlation matrices used in our experiments were created to represent three structurally different situations; for an illustration see Fig. 7.2. The actual matrices will obviously be a bit larger than what is shown here. The three cases are those of uncorrelated random variables, strongly positively correlated variables, and a case where strong positive and negative correlations are mixed. Uncorrelated demand implies that there are no systematic relationships among the demands from our customers. Strongly positive correlations imply that there are some underlying phenomena that drive the demands of our customers, mostly making them all large or all small (although there is a small probability that one is small while another is large). The mixed case represents a situation where we have several sets of customers. Within each set, correlations are strongly positive, while between sets, the correlations are negative. A simple example could be that we have a set of customers experiencing large demands when the weather is good and another experiencing large demands when the weather is bad. But on top of these trends, there are individual variations, and hence the correlations are not ± 1. Negative correlations also occur if a given customer has two alternative production sites (origins), and we know for sure that he will send goods in a given week, but we neither know from which origin nor how much. Also, if a customer is known to send goods in a given week, but the day is uncertain, these two origin-destination pairs will have negatively correlated demand (as they will be seen as different commodities in the model).

Note that the case of all negative correlations does not exist in any interesting way. Three demands cannot all be pairwise negatively correlated unless the correlations are rather small.

In the deterministic case below, we simply use expected demand. The major results are summed up in Table 7.1:

As we can see from Table 7.1, four trucks are needed in all cases. In two cases, even the operating costs are the same. However, this does not imply that the trucks follow the same routes. In fact, the truck routes, as well as the flows for the commodities, are rather different, and that is why the solutions behave very differently, as we shall soon see.

The total costs are given in the rightmost column. All the stochastic cases are rather similar. The deterministic cost is quite a bit lower than the three stochastic cases. However, as this is the optimal objective function value within the

Table 7.1 Results of the four cases solved

The cases	No. of trucks used	Cost of operating the schedule	Expected cost of outsourcing	Total cost
0 correlation	4	4850	396	5246
Mixed case	4	4850	285	5135
0.8 correlation	4	4600	565	5165
Deterministic	4	4700	0	4700

optimization problem and not the true expected cost of using the solution from the deterministic model, this low cost is misleading (as it is normally in deterministic models). We shall return to the true expected cost in a short while. Note that outsourcing was allowed also in the deterministic case.

Use of Outsourcing
Although the total expected costs are similar in the three stochastic cases, the use of outsourcing is somewhat different. With strongly positive correlations, we are a bit modest while setting up the network and rather choose to plan for a somewhat large amount of outsourcing.

Strong Positive Correlations
There are clear differences in the needs in the three stochastic cases. For strongly positive correlations, the probability of all (most) demands being large at the same time is rather high. There will be a few scenarios that really drive the solution. The focus in this case is on facilitating these cases of high demand, and the result is a plan where parts of *really* high-demand scenarios are foreseen as being outsourced, but where most other scenarios are easily accommodated within the given solution. Low costs here come from the fact that the schedule is set up to accommodate the most probable scenarios, and these are scenarios where most demands are high at the same time. Money is not spent creating alternative connections in the network. Rather, one sets up a network that is well suited for the most likely high-demand scenarios, and money is otherwise spent on carefully planned outsourcing.

The need to be flexible in the routing of goods in the network is low, as we are quite close to a worst-case (stochastic) situation, given by all correlations equal to one. Hence, flexibility has taken a back seat relative to the need to accommodate the high-probability, high-flow scenarios.

Mixed and Uncorrelated Cases
In the other two cases, we are much further away from the worst case, and we need a network that can facilitate a wide variety of rather different flows. When the scenarios with some demands high and some low have high probabilities, we need flexibility in the possible ways of routing flow. The reason is that there are many different ways of combining high and low demands, none of them very likely, but the total probability of such combinations is very high. Hence, we invest in flexibility in routing. One way of achieving this flexibility is to use consolidation. When two

Table 7.2 Expected costs resulting from using four different solutions in three different stochastic environments. The numbers in parenthesis are adjusted for the differences in design costs for the networks

	Expected cost of outsourcing		
The cases	0 correlation	Mixed case	0.8 correlation
0 correlation	396 (396)	290 (290)	2141 (2141)
Mixed case	432 (432)	285 (285)	842 (842)
0.8 correlation	1802 (1552)	1776 (1526)	565 (315)
Deterministic case	973 (823)	681 (531)	1402 (1252)

negatively correlated demands can share a path, we easily get a very effective use of capacities if these two flows are consolidated. Hence, we here invest more in routing flexibility and get much less outsourcing. Zero correlations also allows for a certain amount of capacity sharing, although less than that when we have negative correlations.

What If We Are Wrong About the Correlations?

What is as important as looking at how the costs are distributed, is to look at how the solutions perform under conditions different from what they originally were intended for. We take the schedule obtained from solving our problem using a scenario tree based on one correlation matrix and test it using another scenario tree based on the same marginal distributions but now with a different correlation matrix. To do this test, we simply fix the network design variables in our general model and solve only for the second stage. The results are found in Table 7.2.

When looking at the columns in Table 7.2, we note something interesting. The performance of the various solutions is very different when they are tested under conditions they were not intended for. Why is that? We mentioned earlier that for the case of strong positive correlations much of the focus of the design is on facilitating the really difficult (high-demand) cases (as they have high probabilities). A side effect of this is that the schedule is not very flexible with respect to the routing of goods. Hence, when scenarios it thought had very low probabilities (and so were planned to have a lot of outsourcing) turned out to be rather probable, the expected outsourcing costs become quite high. The mixed case knows that many peculiar scenarios can occur. As a result, the design allows for more flexibility in routing, and hence, it handles the other cases much better. The zero correlation case has problems with strongly positively correlated demands, as it does not handle the high-demand cases well (and they now have much higher probabilities than it planned for).

The Most Flexible Setup

Hence, we see that getting the correlations wrong when planning can lead to serious problems later on. We also see that the deterministic case has its difficulties, and in all cases it is the second worst. For this example we see that the mixed case behaves the best on average, as well as having the best worst- case behavior. This is probably caused by it producing a network for handling all types of structures, resulting in a desirable flexibility in routing.

7.6.2 Relation to Options Theory

We may choose to see these results in light of options and option values. As can be seen from Table 7.1, the first two schedules cost the same. Of course, when their assumptions of correlation structures are correct, they are optimal. But consider how they function for the case of strongly positive correlations in Table 7.2. One has a much better expected performance than the other. Hence, it seems that the mixed case has produced options with much higher values than those coming from the zero correlation case. The actual value of these options will of course depend on what is the correct situation. If 0.8 correlations are the correct description, the options from the mixed case are worth $2141 - 842 = 1299$ more than those of the zero correlation case. It is worth noting that since we do not know what the options are, option theory—as it is normally defined—cannot help us find these values. Option theory can only value options that are predefined. Stochastic programming gives us the values directly and, by providing us with optimal decisions, at least can help us try to understand what they are. We shall return to that issue in a few moments.

7.6.3 Bidding for a Job

A major observation from our case is that there certainly exist situations where it is crucial to get the correlation structure right. We also see that if we get it wrong, some settings are still better than others. Flexibility and robustness are important issues here. Some choices of correlations lead to better solutions with respect to the flexibility in the routing of goods. Understanding this process is crucial for practical use. Overlooking correlations totally by resorting to deterministic models is equally dangerous. Based on our findings, it is crucial for an LTL carrier or an airline to correctly estimate co-variation, and it is certainly important to take stochastics into account. Otherwise, a chosen schedule may turn out to be very fragile.

Consider some carriers, independently of mode, who participate in a competitive tender. Assume we have rational participants knowing that uncertainty exists (and who therefore use stochastic models) and that the participants do not face any form of economy of scale/scope. If so, the players would try to win the tender by obtaining the largest possible profit margin. Assume further that these players use the same planning tools and have the same objective (profit) function, so that the only difference between them would be how they perceive the "world." Narrowing it further down so that the only difference between the participants is how they think the demands are correlated, we could still end up with substantially different bids. Using the results from this section, we can clearly observe that a firm believing that the correlations were like the mixed case would win the bid (assuming identical planned profit margins). The only problem is that if the demands turned out to be completely uncorrelated or strongly positively correlated, winners curse would be likely to occur. The winner would lose money. A carrier aware of this problem could

of course make a bid based on a worst-case view on the correlation matrix. That would result in a bid not only suffering from winner's curse but also a losing bid.

7.7 The Implicit Options

So, what is going on when some schedules provide more flexibility in routing than others? What are the options inherited in the solutions? What does a good robust schedule look like? To find that we must study the solutions we found. Even though the solutions obviously contain different options, they are hard to find. This is important to realize. Option theory focuses on well-defined options, like the option to wait or the right to do something specific in the future. Here it is more subtle. You cannot subtract a non-robust solution from a robust one and interpret the difference as an option. The solutions are simply different, and one is better than the other. So, in a sense, it is easy to find option values (and costs), but not to actually find the options. This marks the difference between stochastic programming and option theory. Option theory can only value structures already defined, it cannot find them. Stochastic programming can value total solutions, obviously containing options, but finding what the options are must be done manually. Even so, this puts us in a much better situation than if we had just option theory to work with. We shall now see that for the service network design problem, we have indeed found out (at least to some extent) what characterizes a good robust schedule with flexible routing of goods.

> Options are implicit in the solutions, not explicit on top of another solution.

We shall see that on the one hand the structures are somewhat simple, and on the other hand, deterministic models would not produce these structures. We shall also observe that the well-known structure of hub and spoke, in particular, the idea of consolidation, will be part of the solution structures without being enforced. It is simply optimal, in light of randomness in demand, to use consolidation. Under certain reasonable conditions, consolidation takes place in a hub-and-spoke environment.

7.7.1 Reducing Risk Using Consolidation

Consolidation in LTL trucking is normally thought of as a way to accommodate the fact that most loads are less than one truckload. In deterministic models, consolidation helps keep truck utilization up and also facilitates the use of reasonably large trucks over longer distances. However, there are also risk-related reasons for using

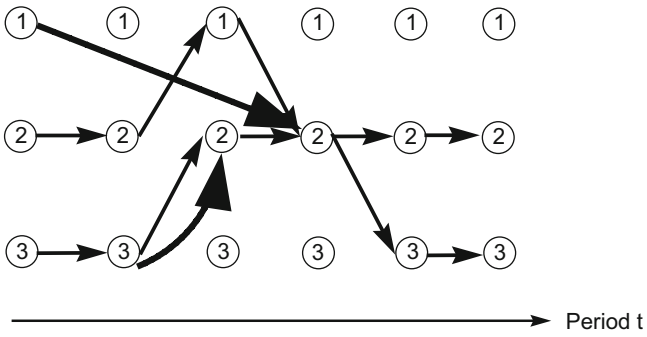


Fig. 7.3 Deterministic problem with no consolidation, using two trucks

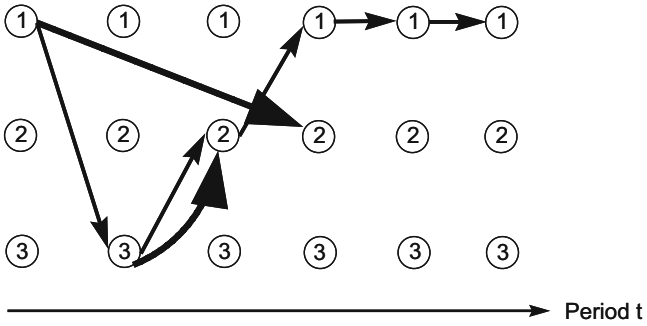


Fig. 7.4 Stochastic problem using consolidation between nodes 3 and 2. Two trucks were used

consolidation. By risk in this case we simply mean the risk of having to reject or outsource demand as that (presumably) is bad for business.

Let us look at a minor example where each load has an expected size equal to the truck capacity. In such a setting, consolidation will not be part of a deterministic solution. Consider the example with two commodities in Figs. 7.3 and 7.4. One commodity becomes available at node 1 at time 0 and has to be delivered to node 2 within three time periods. The other commodity becomes available at node 3 at time 1 and has to be delivered to node 2 within one time period. The thinner arrows in the two figures show the optimal schedule for the two trucks that will be used to transport the commodities from their origins to their destinations. We note that there will not be any consolidation in the deterministic solution (assuming just these two commodities) so that each commodity will use its own truck. Note that the solutions are not unique (but all solutions have the same structure).

In the stochastic solution, we see both trucks following the same path. This is a different and, in the example, slightly more expensive schedule, compared to the deterministic one, as the trucks move more. However, it allows the two commodities to share the joint capacity of the two trucks from node 3 at time 1 to node 2 one period later. This is an example where commodity 1 is sent through an intermediary

breakbulk (node 3), before being consolidated with commodity 2 using the two trucks servicing the link from node 3 at time 1 to node 2 one period later. Having this kind of solution where two commodities share the common capacities of two trucks makes expensive outsourcing less likely compared to if they had used one truck each as in the deterministic case.

Let us illustrate by using some numbers. Assume trucks have capacity 2, and let the demand for each of the commodities be 0, 2, or 4, each with a probability of $\frac{1}{3}$. Hence, in total, there are nine possible scenarios, and if the demands are uncorrelated, each scenario has a probability of $\frac{1}{9}$.

Hedging Using Consolidation

Let us see what happens if we use the deterministic solution of Fig. 7.3 but in fact face this uncertainty. Whatever demand cannot be met will be outsourced. Of the nine scenarios, five will result in outsourcing, and the expected amount outsourced is $\frac{4}{3}$. If we test the schedule in Fig. 7.4, we note that the expected amount of outsourcing will be $\frac{8}{9}$ or $\frac{4}{9}$ lower than for the deterministic schedule. What has happened is that the two scenarios $(4, 0)$ and $(0, 4)$ can now be handled without outsourcing due to consolidation, that is, due to the ability to share transportation capacity.

What we observe here is closely related to what has been observed in other parts of the operations research literature. For example, in the inventory theory, when the number of warehouses/inventories drops, safety stocks can be reduced without reducing the service levels. In finance, the risk in a portfolio of various financial instruments can be reduced by diversification, keeping the expected return the same.

Such use of consolidation as a means to hedge against uncertainty is a feature that has not been described in the service network design literature. There can be several reasons for this, but since the literature almost exclusively refers to or uses deterministic models, hedging against uncertainty becomes irrelevant. This is so despite the fact that most researchers in the field are well aware that uncertainty plays a major role in the problem itself. The models used, however, are deterministic. Notice again that consolidation is not a property we enforce on the solution, but a structure that emerges because it is good for the overall behavior of the schedules.

7.7.2 Obtaining Flexibility by Sharing Paths

In the above example there are only two commodities. Let us now pass to an example with four commodities. We shall see that it is good not only to consolidate but also to have many paths available for each commodity. Also here, all observed aspects of consolidation will come from stochastics and not from standard volume related arguments.

Figure 7.5 illustrates the example. We have four commodities becoming available at their respective origins in the first time period and having to be transported to their

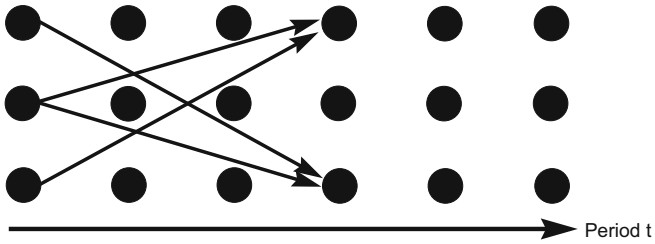


Fig. 7.5 Origin and destination of commodities

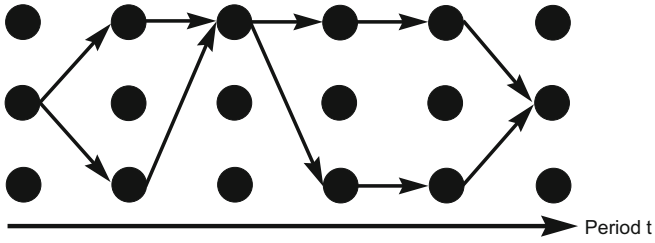


Fig. 7.6 Schedule of trucks based on deterministic demand

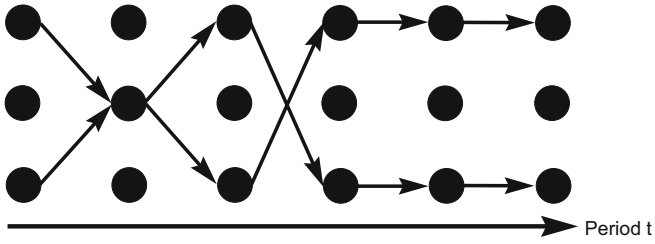


Fig. 7.7 Schedule of trucks based on stochastic demand

destinations within the time indicated by the arrows. This transport can take place either using the company's own trucks or using outsourcing. The schedule in Fig. 7.7 is based on stochastic demand, while for the schedule of Fig. 7.6, demand equals the expected demand from the stochastic case. Otherwise, all parameters are the same (Fig. 7.7).

The deterministic and stochastic schedules require the same number of trucks. In the deterministic case no outsourcing is used (when the demand is deterministic), while the stochastic case uses some outsourcing (less than 0.1% of the total flow in all the scenarios) to be able to transport all the commodities to their destination. (This does not imply, however, that the deterministic solution is better with respect to outsourcing. When the deterministic solution is subjected to random demand, a total of 1.17% of the total flow is outsourced.)

More Flexibility in Routing for Stochastic Case

The most substantial difference between the two solutions is the number of paths connecting the various O–D pairs. In the deterministic case there are two O–D pairs that are connected with one path, while the two others are connected by two paths. In the stochastic case each O–D pair is connected by at least two paths. The higher number of paths in the stochastic solution makes it easier to switch the flow of commodities from one path to another, if required because the first path is taken by another O–D pair. Such situations might occur when there is a surge in demand on a specific O–D pair that fully or partially can be routed on a different path. This larger number of paths in the stochastic solution gives us more operational flexibility when routing commodities through our network. In most cases, this flexibility increases the capacity available to each commodity, without having to increase the total capacity in the network by adding more and/or larger trucks. This makes our stochastic solution perform better under uncertainty.

This result is in line with what we have observed in a variety of test cases. When uncertainty becomes important, our solutions habitually move away from "direct connections" between origins and destinations in deterministic cases, to more hub-and-spoke looking networks where the freight is being shipped through intermediary terminals. This is not because it was "decided" a priori that freight between some O–D pairs should be handled this way, but because it turned out to be the best solution to deal with the uncertainty. Using our model has shown us that consolidation in hub-and-spoke networks takes place not necessarily due to economy of scale or other similar volume-related reasons, but as a result of the need to hedge against uncertainty.

So, the conclusion so far is that a good solution will try to provide many paths for each O–D pair, such that capacity is shared with other (different) O–D pairs on each of these paths. This provides optimal operational flexibility. It implies that as soon as some demand is low, others can immediately utilize the freed capacity if they are in need. A deterministic model would never produce such structures.

7.7.3 How Correlations Can Affect Schedules

Look back at Figs. 7.3 and 7.4. When we calculated the value of using the stochastic rather than deterministic model, we assumed that the demands were independent. Assume instead that the two demands are perfectly negatively correlated. In that case, the sum of the two demands will always be 4, and the expected outsourcing will be zero. But if we use the solution from the deterministic model, expected outsourcing will remain at $\frac{4}{3}$. What we observe is not surprising, but important. Negative correlations imply a chance to achieve hedging, but only if the two negatively correlated flows can be set up so as to share capacity. And the more negatively correlated they are, the more important the issue is.

Always look for negative correlations.

On the other hand, in the same example, if the two demands are perfectly positively correlated, the expected outsourcing will be the same for the deterministic and stochastic solution, since, in fact, consolidation will never take place.

More generally, as long as random variables are not perfectly positively correlated, there is something to be gained from flexible routing. The more we move toward perfectly negative correlations, the higher is the potential for hedging. As flexibility normally comes at a cost, there is a trade-off between the cost of achieving operational flexibility and the expected gain from the investment. Of course, when there are many random variables, the relationships are more complex. But even so, we can conclude as follows: Operational flexibility is achieved by having many paths available for each O–D pair, such that each of these paths is shared by other O–D pairs. The more negative the correlations are, the more there is potentially to be gained from well-structured schedules.

7.8 Are Deterministic Models Useless?

A lot of research has gone into stochastic network design. One interesting question that has been asked a few times is: Even if the deterministic solution is bad—could it be useful? It turns out that the answer is often yes, and it is useful. This result is based on numerical tests rather than theoretical considerations. But even so, the reasoning is useful.

The most important observation is that a major reason why deterministic solutions based on expected demand are bad is that the arc capacities are set far too low. In the resulting network, there is close to a 50% chance that in the stochastic setting, some demand must be rejected. Independently of the network structure, which is just too conservative, much more capacity is needed.

This is close to a general observation. When capacities (of any type) are to be set, they will be too low if the expected demand is used. So an obvious choice, even if you stay deterministic, is to use something larger than the expected demand, more like the 70th or 80th percentile. Out-of-sample tests will then show what is the best choice.

In addition, some other tests can be made. The first is to let the deterministic model set which arcs to open and then fix them open (and the rest closed) in the stochastic model. The stochastic model will then still be a two-stage stochastic model, but where there are no longer binary variables in the first stage, just continuous capacities. So this is *much* easier to solve. In many cases, the network structure from the deterministic model is equal to or almost equal to the one from the stochastic model. So, the weakness in the deterministic solution really comes

from the capacities. If this is the case—and that must be tested, not assumed—a heuristic for the overall problem is to first solve a deterministic network design problem to fix the binary variables and then a two-stage stochastic linear program to set capacities and flows. But how do you test if this works? After all, that requires solving the original stochastic model. Well, this chapter has indicated the answer; solve small cases, try to understand what is going on, and argue, probably verbally, why the results scale (or not). We believe the results in this chapter are reasonably scale. Why should sharing paths and having many paths for each commodity only be valid for small cases? As we are talking about finding good solutions to a very hard problem, we should be willing to cut some heuristic corners here and there.

Another result that has emerged is that the deterministic network structure is contained in the stochastic one. In that case—again this must be tested not assumed—we can fix the corresponding binary variables to one but leave the rest free. The remaining stochastic model then still has some discrete variables, but much fewer, as many arcs are fixed open following the deterministic solution. Certainly, the remaining problem is of the same type as the original one, so this may not help in all cases. However, apart from providing information about the problem itself (why did this happen?), this can be algorithmically useful. If you use a heuristic that builds a solution (however, you do it), you can start from the deterministic solution instead of starting from scratch.

Again, by all means, do not *assume* that this will work. It certainly might not. But for whatever problem you are studying, it is worth testing if the deterministic model could be "bad but useful." It may not be as easy as splitting between discrete and continuous, but maybe there is a subset of variables that are set well or correctly by the deterministic solution. Test is needed to see if it works. And remember, this is not just done to facilitate solution procedures, and it also provides information about the problem itself—why did this happen?

> Deterministic solutions can be useful even if they are bad.

7.9 Conclusion

The purpose of this chapter has been to illustrate how the ideas in this book can be used to analyze a new problem. The setting here is that of a graduate student studying the effects of uncertainty on a discrete optimization problem. We trust other students can do the same with other problems. Note the need to use proper scenarios from Chap. 4 to make sure that you analyze what you think you are analyzing.

Chapter 8
A Multi-dimensional Newsboy Problem with Substitution

With Hajnalka Vaagen

I suffer from a severe fashion disorder.
– *Cosmo Fishhawk*

Fashion is a form of ugliness so intolerable that we have to alter it every six months
– *Oscar Wilde*

In this chapter you will meet a problem which is inherently two-stage. The first inherent stage is to decide on production levels for a number of substitutable products with correlated demands. This is followed by a second stage where demand is met, partly by giving customers what they want and partly by giving them acceptable substitutes. The chapter is based on [83].

You shall see that the model, even if used in a full-scale industrial setting, is small, with negligible CPU time. However, this does not mean that the problem is straightforward. The modeling itself is original, and maybe most importantly, the distributions are complicated. So to solve this problem satisfactorily, we shall need to call upon results from Sect. 4.4.2. So not all stochastic programs are hard because they are large. Complexity may also come from modeling as such and from complicated distributions. We shall see how simplifying assumptions on the distributions will cause severe losses of profit.

Stochastic programming offers the ability to handle complicated distributions. Difficulties do not always stem from size.

A. J. King, S. W. Wallace, *Modeling with Stochastic Programming*, Springer Series in Operations Research and Financial Engineering,
https://doi.org/10.1007/978-3-031-54550-4_8

8.1 The Newsboy Problem

Let us first review the classical newsboy problem. It is an interesting stochastic optimization problem in its own right. Imagine a boy selling newspapers at a street corner in a big city. Each morning he goes over to the newspaper headquarters to buy a number of newspapers at a unit cost of c. He then goes to his street corner. For each paper he sells, he gets v, while each paper not sold gives him a salvage value (maybe recycling) of g. He has been selling for a long time, so he has a good feeling for the demand, and it can be described by a scenario set d^s for $s \in S$. So in this problem, he can order newspapers only one time. That is a crucial aspect of the newsboy problem.

> The newsboy problem is about perishable products where all production must take place before demand becomes known.

Let

x be the number of newspapers bought.
y^s be the number of newspapers sold in scenario s.
w^s be the number of newspapers not sold in scenario s.
c be the purchasing price for a newspaper.
v be the selling price.
g be the salvage value for an unsold newspaper.

The problem is then

$$\begin{aligned} \max \quad & \sum_{s\in S} p^s(vy^s + gw^s) - cx \\ \text{such that} \quad & \begin{cases} y^s \le d^s & \forall s \in S \\ w^s = x - y^s & \forall s \in S \\ x, y^s, w^s \ge 0 & \forall s \in S \end{cases} \end{aligned} \tag{8.1}$$

Note that $y^s \le x$ is implied by the other constraints.

This problem has an analytical solution, but we shall not worry about that now. It can also be solved numerically, of course, and the extension presented in this chapter can only be solved numerically.

The problem we are facing in this chapter extends the classical newsboy model in two directions. First, we have several *items*, as we shall call them, not just one. These will all represent the same basic need for the customer. Think of high-tech mountaineering jackets for men, in different colors and designs. These are typically

fashionable in only one season, and the production lead times are so large that all production decisions must be made before demand is realized. The salvage value corresponds to selling on sale or throwing away. Since these items satisfy the same basic need, their demands are strongly dependent. We shall return to how to describe the demand distributions.

Note that this is a newsboy-like problem by *construction*. The products are made at specialized facilities in China and sent to the main markets by ship. Had the manufacturer decided to produce closer to the markets or sent the products by plane, we would not have faced a newsboy-like planning problem. But that is not done as it would be too expensive.

The other extension in this model is that we shall allow product substitution. That is, if you come in wanting a jacket of a certain color and design but can only get the right color, but another design, there is a certain chance that you will accept a substitute. And the pattern of substitution is as complex as the demand distributions, but we will return to that in a while as well.

The actual setting of the model is not that of a retailer and a customer like you, but that of a manufacturer and its retailers. In this context substitution will imply that the manufacturer does not deliver a retailer what he wants, but an acceptable substitute or nothing at all. So only a certain percentage of those that are denied their first choice will accept a given substitute. This is called manufacturer-directed substitution and is different from consumer-directed substitution, which is much more difficult to model.

8.2 Introduction to the Actual Problem

Capturing market trends and satisfying customer demand by supplying quality products in a very short time is the dominant challenge in modern manufacturing. Sophisticated information technology enables fast and accurate information flow. However, when information available at the time of planning is largely qualitative and based on trend estimates and lead times are pressed to a minimum (but still very large), it is crucial to know what kind of information to look for, how to interpret it, and how to apply it in portfolio and production planning. Obviously, mapping the true problem complexity and focusing on the quantitative aspects are crucial for creating good solutions. Furthermore, worried over the variety explosion in contemporary markets and its negative impact on own supply chain activities, many suppliers and retailers turn from offering higher variety to a more efficient assortment strategy.

In this chapter we discuss the quantitative aspects of assortment planning in quick response supply chains; sports apparel is taken as the case of analysis. Despite the fact that the assortment planning field is rather new, the motivation to study the problem is high, and much published work can be found; see [58] and [62]

for extensive reviews. Most of this work focuses on analytical formulations of assortment optimization. Different heuristics and strong assumptions on the nature of demand patterns and inter-item dependencies are applied to achieve solutions. Discussions on the numerical complexity, tractability, and applicability of these formulations to real-life problems, as well as empirical tests of the theoretical predictions, are rather vague. The potentially enormous academic contribution in adding rigor and science to the retailers' developed practices is emphasized in [58], much as it has been done in areas like finance. We shall see that a numerical stochastic program can bring results not available via analytical models because of the ability to handle complicated distributions.

8.3 Model Formulation and Parameter Estimation

Based on the recognition that multi-dimensional newsboy models with complex dependencies are analytically difficult to deal with, we analyze whether a numerically based stochastic programming formulation provides more useful solutions. We avoid using heuristics, thereby allowing for complex distributions, correlations and substitution patterns. It is obvious that we are dependent on data; data that cannot be obtained without internal understanding of the specific industry. However, we do not consider this to be a disadvantage, but rather recognition and acceptance of reality.

We assume cost and selling prices rather homogeneous within the group of substitution. The trend driver attributes, on the other side, are qualitative and heterogeneous. When demand becomes known, products are assigned to customers so as to maximize manufacturer's profit. Optimization happens in one stage, with allocation between direct and substitution sales. The optimal solution implies then that the manufacturer controls these values; in other words, it tells the retailers how much of the first and lower preferences they may buy. For a real-life situation with characteristics similar to this problem, we refer to the fashion and sports apparel franchises.

8.3.1 Demand Distributions

When the major decisions have to be made, that is, which items to include in the portfolio and how much to produce, only limited information is available about demand. The reason is that we are facing a product affected by fashion, and it is not known what will be "hot" in the coming season. However, much is known about the structure of demand. Some items are known to be in competition with each other. For example, it might be known that if navy is a popular color, then black will not be popular. Or it could be the other way around with black popular and navy not.

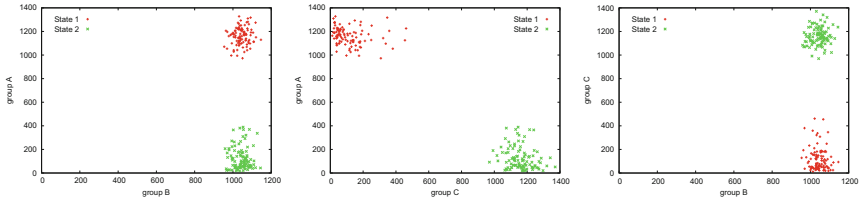


Fig. 8.1 Joint distributions for pairs of item groups as a function of the state of the world

This will be expressed by negative correlations in demand. Positive correlations will occur for colors that are popular (or unpopular) together. In one season all clear colors may be popular, while all Earth colors are not. So based on such knowledge correlations can be estimated. Some items will be more stable—such as white—and are not so much affected by fashion. The same type of arguments can be made about design.

But there is more to it than that. Assume we have three groups of items. The items in Group A are popular with a certain probability, while those in C are popular when those in A are not. We say that the world is in State 1 if those in A are popular and in State 2 otherwise. We also have a set of items that have a stable demand. We call this group B. Figure 8.1 shows typical joint distributions for pairs of these groups for a case where the probability of each state of the world is the same, and all scenarios shown have the same probability. So the demand described here is over all items in the group.

Note how each marginal distribution is bimodal and how the bimodality is distributed over only two quadrants when we study two items at a time. This kind of distributions is impossible to handle in analytical models. But as we shall see in this chapter, to get good results, we have to take the bimodality into account. It is also worth noting that *given* the state of the world the distributions are not particularly complicated and can, in fact, be described well with simple log-normal distributions. (We like to avoid normal distributions since they have positive probabilities for negative demand.) We use tools from Sect. 4.4.2 to create these discrete distributions.

> Numerical models can handle complex distributions, even multi-modal distributions, using the tools of Sect. 4.4.2.

To define the state-conditioned distributions, we use aggregated demand data across whatever will become popular/unpopular. Hence, instead of approximating the overall distribution, we generate scenarios for each state of the world independently, using scenario generation tools from Sect. 4.4.2. The overall distribution is then built by connecting all the scenarios by state probabilities (probabilities

that sum up to one). Under limited information, the two states occur with equal probabilities. This way, the uncertainty around the individual items' popularity is captured.

8.3.2 Estimating Correlation and Substitution

Correlation and substitution measures express the dependencies among the individual items. Assortment planning in our quick response settings (fashion and sports apparel) largely happens when the information is limited to qualitative knowledge of the trend drivers and aggregated estimates.

Here is how you may think: The existence of two states of the world for some items is described by strong negative correlations in demand for all pairs of competing items. Similarities on trend driver attributes, like the existence of specific technical features across items, define positive correlations. We connect correlations and substitutions by a common information base and describe product substitution based on the grade of similarity with regard to the trend driver attributes. We define the decision independent *a priori substitutability* $\alpha_{ij} \in [0, 1]$, indicating the portion of customers willing to replace item j with item i. We are assuming that the manufacturer, although being able to decide which customers receive their first choice and which are offered a substitute, does not know which specific customers will accept a substitute. Hence, α_{ij} can also be viewed as the probability that a customer will accept the substitute. We assume the n items offered to be each other's potential substitutes with heterogeneous substitutability values. Note the distinction between a priori substitutability and the true substitution, called *factual* substitution, this latter being a decision-dependent outcome of an optimization process, constrained by unsatisfied demand and availability.

For the connection between correlation and substitutability values, consider the following example. The supplier is to make assortment decisions on two identical models in colors black and navy. Assume the information available for decision-making to be:

(a) Color is a strong trend driver, and only black or navy will become popular; this can be described by a strong negative correlation between their demands, (for example -0.5).
(b) Due to the similarity with regard to design, if one becomes popular and faces stock-out, it can partially be substituted by the available one (say, with substitutability of 0.2).

Given the "competition" between the products, it is unrealistic to assume high substitutability. Customers looking for the popular black will likely visit other stores/suppliers to get their preferred choice.

8.4 Stochastic Programming Formulation

We model the following process. In the first step, most appropriately in the design phase, the manufacturer defines the similarity among the products with regard to the trend drivers, hence the substitutability matrix. It is normal that $\alpha_{ij} \neq \alpha_{ji}$. While the safe color white may be an acceptable substitute for the more trendy pink, people wanting white are not very likely to accept pink as they want a subdued look.

Next, based on knowledge of demand, the manufacturer describes marginal distributions conditioned on the state of the world and correlations. Finally, they assess probabilities of the state of the world. If they know nothing, the probabilities are set at 50% each.

Given substitutability and demand distributions, the manufacturer decides the optimal assortment to offer: products to include in the portfolio and their inventory levels. Finally, when the actual retailer demand becomes known, the manufacturer assigns first and substitutes preferences so as to maximize expected assortment profit, given the initial inventory levels and substitutability matrix. The outcome of this process is the factual substitution.

A simple two-stage stochastic program is formulated: The first stage consists of the production decisions, and the second stage (after demand has been observed) optimally allocates direct and substitution sales.
Sets:

S—set of demand scenarios
I—set of items in the reference group portfolio

Variables:

x_i—production of item i
y_i^s—sale for item i in scenario s
z_{ij}^s—substitution sale of item i, satisfying excess demand of item j in scenario s
zt_i^s—substitution sale of item i, satisfying excess demand from all j in scenario s
w_i^s—salvage quantity for item i in scenario s

Parameters:

d_i^s—demand for item i in scenario s
p^s—probability of scenario s
v_i—selling price for item i
c_i—purchasing cost for item i
g_i—salvage value for item i
$\alpha_{ij} \in [0, 1]$—substitutability probability, the probability that item j can be replaced by item i

$$\max \quad \sum_{s\in S} p^s \sum_{i\in I} \left(-c_i x_i + v_i y_i^s + v_i z t_i^s + g_i w_i^s\right)$$

$$\text{such that} \quad \begin{cases} y_i^s + \sum_{j\in I; j\neq i} z_{ji}^s \leq d_i^s & \forall i \in I; s \in S \\ z_{ij}^s \leq \alpha_{ij}(d_j^s - y_j^s) & \forall i, j \in I, i \neq j; s \in S \\ z t_i^s = \sum_{j\in I, j\neq i} z_{ij}^s & \forall i \in I; s \in S \\ w_i^s = x_i - (y_i^s + z t_i^s) & \forall i \in I; s \in S \\ x_i \geq 0 & \forall i \in I \\ y_i^s, z t_i^s, w_i^s \geq 0 & \forall i \in I; s \in S \\ z_{ij}^s \geq 0 & \forall i, j \in I, i \neq j; s \in S \end{cases} \tag{8.2}$$

This maximizes expected assortment profit from ordinary sales, substitution sales, and salvage. The first constraint set states that the total sales for item i—coming from primary demand for i plus all j sales generated by unmet demand for i—are constrained by the total demand for item i. This constraint can be reorganized as

$$\sum_{j\in I, j\neq i} z_{ji}^s \leq d_i^s - y_i^s \qquad \forall i \in I; s \in S \tag{8.3}$$

stating that substitution sales from item i cannot exceed available unsatisfied demand for i. The next constraint set shows the upper bound on the substitution sale of item i for item j, that is, the excess demand for item j with a given substitutability probability α_{ij}. Note the interpretation here. The substitution is done in terms of averages, which is necessary since we do not model the individual customers. After having decided who gets their primary wishes satisfied, the manufacturer will, on average, observe that the portion α_{ij} of those wanting item j, but not getting it, will accept item i. We are assuming that at this point the manufacturer can, in fact, find out which ones are willing to take item i instead of j. They are then offered item i, provided there is something to offer according to the first constraint. We are on the other hand not assuming that the manufacturer knows up front who will accept substitutes, since the retailers are unlikely to reveal such information as it will clearly be used against them in the first round of allocation.

The third constraint set gives the overall substitution sale i from all j's. Next follows the salvage quantity, the quantity of item i left after satisfying primary demand and substitution demand from all j. Finally, non-negativity constraints are given. These constraints imply that the substitution sale of item i is limited to the remaining supply of the item; that is, $z t_i^s \leq x_i - y_i^s$.

8.5 Test Case and Model Implementation

We analyze a real assortment problem with 15 items, from a leading brand name sportswear supplier. Here we attempt to avoid simplifications on the uncertainty and dependencies. The variants within the group are distinguished by the trend driver attributes, style and color. We study whether mis-specifying substitutability, marginal distributions, and correlations has significant effects on the portfolio structure and its profit.

The true situation of this real assortment problem is the one of bimodally distributed individual item demands for about half of the portfolio and dependencies expressed by a correlation matrix with heterogeneous values. This case is denoted Bimodal. For further details see [83]. To say something about the effects of mis-specifying the uncertainty and dependencies, we define additional situations, incorrectly assuming uni-modality in demand distributions. Precisely, log-normality is used here, with means and variances as in the bimodal distributions. We analyze this under three different dependency patterns: a correlation matrix with all entries equal to zero, a correlation matrix with all entries equal to 0.5, and the true matrix with heterogeneous values. The incorrect uni-modal cases are denoted LogN-$c = 0$, LogN-$c = 0.5$, and LogN-$c =$ true, see Table 8.1.

Profit and production levels are analyzed while varying a homogeneous substitutability in $[0, 0.5]$, in addition to using the true matrix (denoted mix). Average substitutability values over 0.5 are unrealistic across a group of 15 items. Note that the test cases with the true correlation matrix (Bimodal and LogN-$c =$ true), having both negative and positive values, are not well suited to directly conclude on the effect of correlations. To do this, we compare the test results of LogN-$c = 0$ and LogN-$c = 0.5$.

8.5.1 Test Results

The expected profit and corresponding production levels for varying substitutability, under the four distributional assumptions, are given by Figs. 8.2 and 8.3. The main point here is how different this production planning problem looks in terms of profit and production depending on the assumptions made on demand and substitution.

Table 8.1 Test cases

Subcase	Marginal distributions	Correlation values c
Bimodal	Bimodal	True matrix
LogN-$c = 0$	Uni-modal	Assumed zero correlations
LogN-$c = 0.5$	Uni-modal	Homogeneous values $c = 0.5$
LogN-$c =$ true	Uni-modal	True matrix

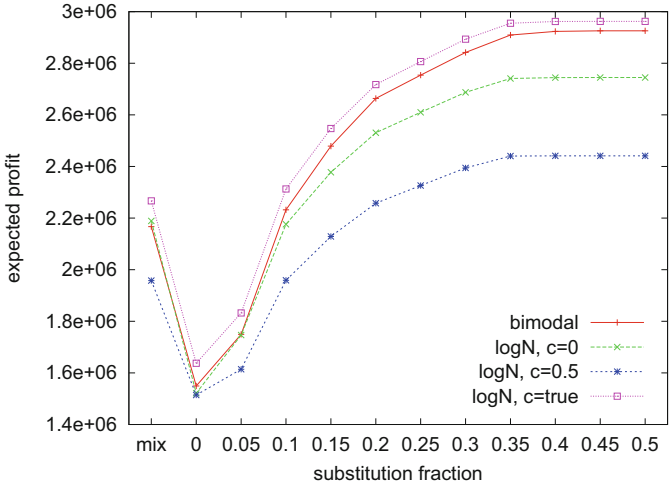


Fig. 8.2 Expected profit versus substitution under the four distributional assumptions

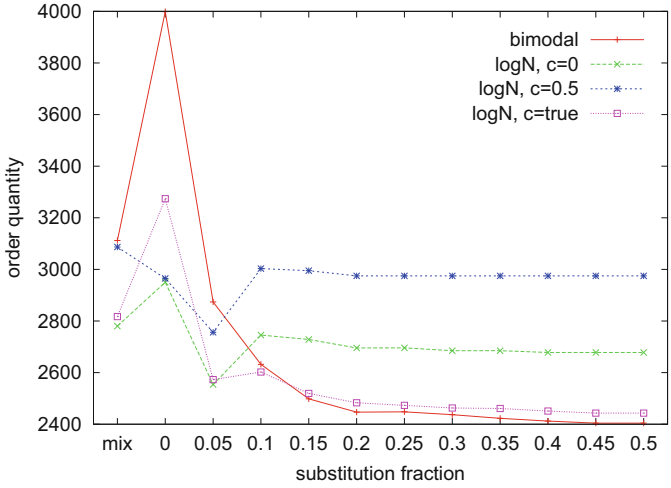


Fig. 8.3 Expected sales versus substitution under the four distributional assumptions

Although these results confirm previous qualitative findings, earlier findings say nothing about the effect of mis-specifying distributions and dependencies. For this, decisions based on incorrect assumptions must be measured relative to the true distributions and dependencies.

The Effects of Mis-specifying Substitutability

Here we analyze the effects of mis-specifying substitutability, replacing the true substitution willingness (mix matrix) by a matrix with its average (0.15) in each

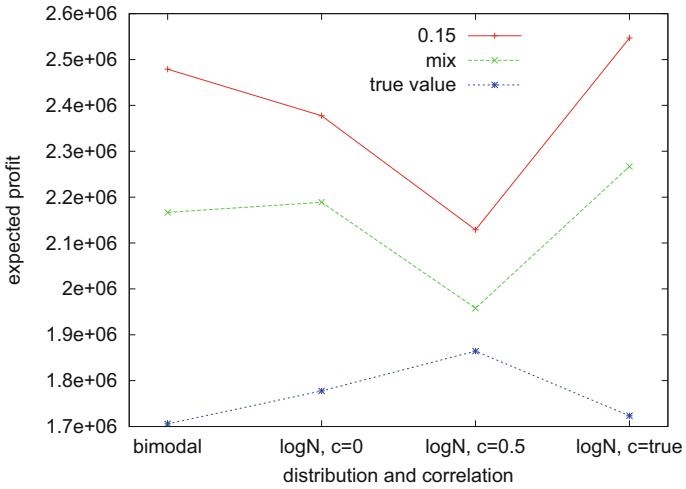


Fig. 8.4 Expected profit under the mix substitutability matrix and its average 0.15, evaluated for the four test cases. The lower curve shows the true profit when the average substitutability of 0.15 is assumed, but the world is described by the Bimodal test case and the mix matrix

element. For all test cases, the expected profit under the average substitutability (as measured within the model) is found to be higher than the expected profit using the mix matrix; see Fig. 8.4. The true Bimodal case results in almost 15% higher expectation under the average substitutability than under the mix matrix (2,478,901 versus 2,166,684).

However, this does not provide the true picture. If production decisions correspond to the average substitutability, but the true substitution willingness describes the world, decision-makers will end up over 30% below their expectation (**1,705,694** versus **2,478,901**). Low production levels under average substitutability (2498 versus 3112 units; Fig. 8.5)—implying low flexibility to adapt to changes when the truth turns out to be the mix substitution matrix—explain the large error in expectations. The substantially lower error (12%) when comparing LogN-$c = 0.5$ with Bimodal is due to the strong positive correlation among the items. Substitutability cannot truly be leveraged on, as the products mostly face stockout or overproduction simultaneously. The optimal plans suggest almost equally high production levels (2995 versus 3087) and, hence, we observe the reduced profit loss. The effects of substitution, and that of mis-specifying it, are less significant when the products are strongly positively correlated.

The Effects of Mis-specifying Distributions and Correlations

Figure 8.6 illustrates the effects of mis-specifying distributions, by incorrectly assuming uni-modality when the world is described by bimodal distributions. We evaluate production decisions obtained from the uni-modal cases LogN-$c = 0$, LogN-$c = 0.5$, and LogN-$c =$ true within Bimodal. Profit loss, then, is evaluated by

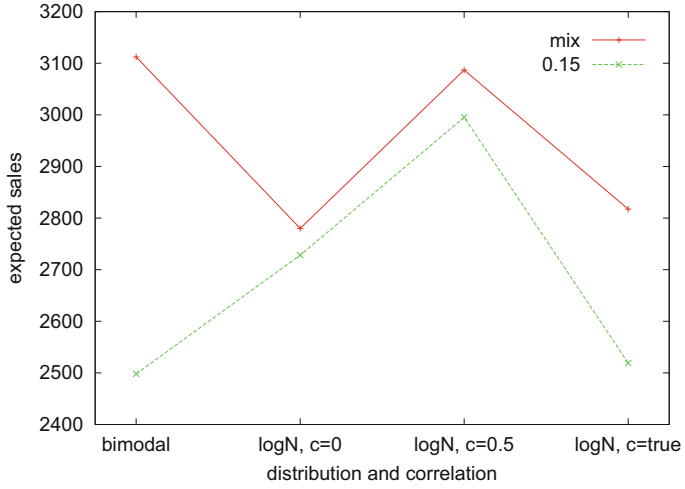


Fig. 8.5 Order quantity for the for test cases, under the assumptions of the mix matrix and its average 0.15

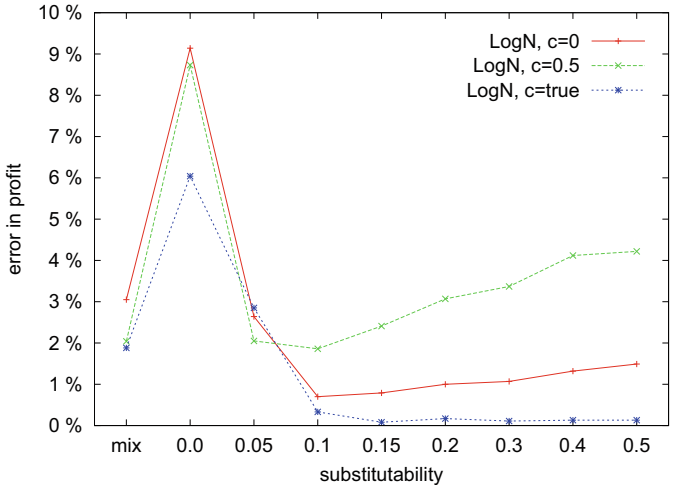


Fig. 8.6 Minimal profit error versus substitution for the uni-modal test cases measured within the true Bimodal test case

comparing the results with the optimal solution of Bimodal, for the corresponding substitution values.

We see, again, that substitution partially compensates for lack of information. When there is no substitution in the group, the error is the lowest when the true correlation patterns are captured, i.e., when LogN-c = true is used. The heterogeneous correlation values and especially the negative correlations imply

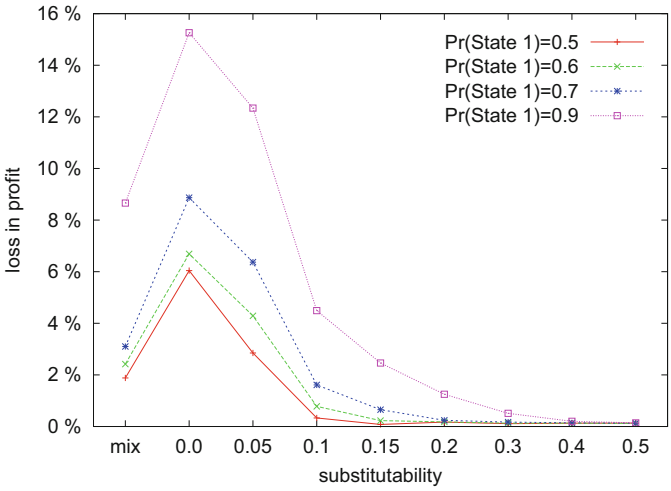


Fig. 8.7 Percentage loss in expected profit under test case LogN-c = true versus Bimodal—evaluated for increasing belief in State 1 for different substitution values

some hedging (see also production decisions for LogN-c = true in Fig. 8.3), contributing to reduction in error when Bimodal actually describes the true world.

The error above is studied under maximal uncertainty, assuming that the two states of the world describing the demand uncertainty occur with equal probabilities. In the next step, we measure the negative effects of incorrectly assuming uni-modality as information about the items' popularity is revealed. In other words, loss in expected profit is evaluated while increasing the belief about State 1 (probability (State 1) $>$ 0.5), for the uni-modal cases LogN-c = true and LogN-c = 0. Production decisions are re-optimized for all cases at hand and for all information levels investigated. The results are summarized by Figs. 8.7 and 8.8. Figure 8.8 gives the profit loss when, in addition to the incorrect uni-modal assumption, correlation values are also incorrect (assumed to be zero). Although substitution partially compensates the negative effects of mis-specifying distributions and correlations, Fig. 8.7 shows that, even with very accurate information (probability (State1)= 0.9) and even when the true correlation matrix is used, distributional assumptions are important; the error is up to 16%.

Resorting to simplifications in distributional assumptions in order to obtain analytical results can lead to very bad decisions.

This analysis, hence, emphasizes the importance of the underlying distributional assumptions when developing forecasting/planning tools for practical implemen-

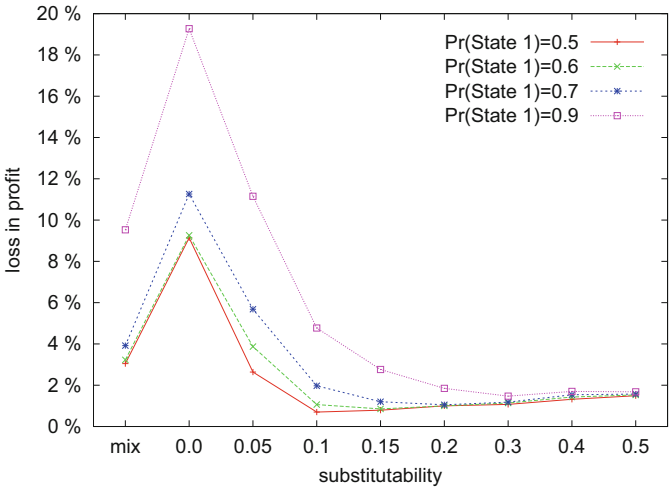


Fig. 8.8 Percentage loss in expected profit under test case LogN-$c = 0$ versus Bimodal—evaluated for increasing belief in State 1 for different substitution values

tation, even when technology and supply chain flexibility allow for continuous information and production updates. To be able to use complicated distributions, we relied on the results of Sect. 4.4.2.

The Substitutable Portfolio Structure

The previous sections provided important insights into the substitutable assortment's sensitivity with regard to the values of crucial demand driver parameters. Here we show the numerical formulation's usefulness when defining the final portfolio structure. Table 8.2 gives the individual item production quantities for the true bimodal case when varying substitutability. For better visualization, production levels under 20 units are eliminated. We observe that the optimal portfolio profit implies trimming some of the products. These products, individually, contribute to the performance and are initially included in the portfolio. However, their substitutability to other products makes them redundant. Observe also the substantial difference in decisions under true substitutability (mix) and when assuming the homogeneous average of 0.15 across the group.

The results in this chapter are potentially important in assortment planning and could not have been achieved by existing analytical formulations. These models have limited potential in even detecting complex effects of mis-specifying uncertainty and dependencies.

Our case provides empirical results on how the true problem complexity affects the outcome of the planning models. Particularly, we show the internal uncertainty introduced by commonly accepted and applied simplifications on uncertainty and dependencies among the items. In fashion and sports apparel, as well as in other related areas with strong uncertainty about which items become popular,

Table 8.2 Changes in production quantities for different substitutability values

Subst.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Total
mix	93	579	486	56	197	84	197	243	194		172	784			21	3105
0	389	393	393	152	152	152	150	153	153	150	389	347	334	334	358	3999
0.05	354	384	378	161	160	162	167	163	163	52	402	181	45	43	59	2874
0.1	220	410	338	102	111	120	299	190	190		179	311			161	2632
0.15	327	426	424				317	115	123		274	376			115	2498
0.2	280	512	508				333				314	468				2435
0.25	155	627	619				249	26			207	561				2444
0.3	79	732	734				128				100	638				2411
0.35		833	832				39				32	658				2393
0.4		839	852									681				2372
0.45		810	875									699				2383
0.5		801	831						21			752				2404

assuming uni-modality in demand distributions seriously affects the quality of the planning. The effects of these general underlying assumptions are significant even when it is possible to continuously update information and production decisions. This understanding is potentially important with regard to evaluating appropriate planning software available for industrial applications. Underlying assumptions in these tools are largely invisible for practitioners.

8.6 Conclusion

This chapter illustrates the value of using a numerically simple stochastic programming formulation of the multi-item substitutable newsvendor problem. Existing analytical formulations are general in their findings; however, solution heuristics and simplifications on the dependencies are frequently applied, and hence it is not at all clear how to read the results.

The stochastic program is simple and handles real problems of substantial size. The CPU times are negligible. As such, it is potentially useful in practical Applications, particularly fitted to treat the complex nature of demand uncertainty and dependencies observed in quick response industrial environments, such as fashion and sports apparel. The presented approach is an appropriate decision support tool in assortment planning, not just in defining optimal inventory levels but also suggesting structural changes where appropriate, such as product line trimming. This is clearly important when the negative effects of variety explosion are substantial for suppliers/retailers and when the focus is changing from increasing variety to satisfy heterogeneous customer needs to defining more efficient portfolio planning strategies.

We analyzed how different parameter values affected the assortment planning problem and hence the importance of being able to include them in the models. Particularly, and similarly to financial problems, we observed the impact of dependency patterns in a product portfolio. That said, we pointed out that the error of misspecifying distributions and dependencies is large. The value of substitution is high and compensates, to some extent, for lack of information; however, only given that the true substitution pattern is captured and quantified by the model. We found up to 30% increase in expected profit when applying our assumed true substitutability matrix versus the common simplification of using average values. The difference stems from structural differences in the first-stage inventory/production levels. We also showed that even when technology and supply chain flexibility allow for continuous information and production updates, the underlying distributional and dependency assumptions used in the planning models are crucial. We believe this observation can be useful in developing decision support tools for industrial use.

References

1. Laurent El Ghaoui Aharon Ben-Tal and Arkadi Nemirovski. *Robust optimization*. Princeton Series in Applied mathematics. Princeton University Press, 2009.
2. David R. Anderson, Dennis J. Sweeney, Thomas A. Williams, Jeffrey D. Camm, James J. Cochran. *An Introduction to Management Science: Quantitative Approaches to Decision Making*. West Publishing Company, 1990.
3. Sebastián Arpón, Tito Homem-de Mello, and Bernardo Pagnoncelli. Scenario reduction for stochastic programs with conditional value-at-risk. *Mathematical Programming*, 170(1):327–356, 2018.
4. M. Ball, C. Barnhart, G. Nemhauser, and A. Odoni. Air transportation: Irregular operations and control. In C. Barnhart and G. Laporte, editors, *Transportation*, number 14 in Handbooks in Operations Research and Management Science, chapter 1, pages 1–67. Elsevier, 2007.
5. Güzin Bayraksan and David P. Morton. Assessing solution quality in stochastic programs. *Mathematical Programming*, 108(2–3):495–514, Sep 2006. https://doi.org/10.1007/s10107-006-0720-x.
6. Güzin Bayraksan and David P. Morton. A sequential sampling procedure for stochastic programming. *Operations Research*, 59(4):898–913, 2011. https://doi.org/10.1287/opre.1110.0926.
7. Dimitris Bertsimas and Nishanth Mundru. Optimization-based scenario reduction for data-driven two-stage stochastic optimization. *Operations Research*, Apr 2022. https://doi.org/10.1287/opre.2022.2265.
8. Dimitris Bertsimas and Melvyn Sim. The price of robustness. *Operations Research*, 52(1):35–53, 2004.
9. John R. Birge and François Louveaux. *Introduction to Stochastic Programming*. Springer-Verlag, New York, 1997.
10. İ. Esra Büyüktahtakın. Scenario-dominance to multi-stage stochastic lot-sizing and knapsack problems. *Computers & Operations Research*, 153:106149, 2023. ISSN 0305-0548. https://doi.org/10.1016/j.cor.2023.106149. https://www.sciencedirect.com/science/article/pii/S0305054823000138.
11. Sheng-I Chen and Delvinia Su. A multi-stage stochastic programming model of lot-sizing and scheduling problems with machine eligibilities and sequence-dependent setups. *Annals of Operations Research*, 311(1):35–50, Apr 2022. ISSN 1572-9338. https://doi.org/10.1007/s10479-019-03462-1.

A. J. King, S. W. Wallace, *Modeling with Stochastic Programming*, Springer Series in Operations Research and Financial Engineering,
https://doi.org/10.1007/978-3-031-54550-4

12. Cong Cheng and Lixin Tang. Robust policies for a multi-stage production/inventory problem with switching costs and uncertain demand. *International Journal of Production Research*, 56 (12):4264–4282, 2018. https://doi.org/10.1080/00207543.2017.1413257.
13. Hanjun Dai, Yuan Xue, Zia Syed, Dale Schuurmans, and Bo Dai. Neural stochastic dual dynamic programming, 2021.
14. George B. Dantzig and Gerd Infanger. Large-scale stochastic linear programs—importance sampling and Benders decomposition. In *Computational and applied mathematics, I (Dublin, 1991)*, pages 111–120. North-Holland, Amsterdam, 1992.
15. T. Homem de Mello and B.K. Pagnoncelli. Risk aversion in multistage stochastic programming: A modeling and algorithmic perspective. *European Journal of Operational Research*, 249:188–199, 2016.
16. Paul Dommel and Alois Pichler. Foundations of multistage stochastic programming, 2021.
17. J. Dupačová, N. Gröwe-Kuska, and W. Römisch. Scenario reduction in stochastic programming: An approach using probability metrics. *Mathematical Programming*, 95(3):493–511, 2003. https://doi.org/10.1007/s10107-002-0331-0.
18. Jitka Dupačová and Werner Römisch. Quantitative stability for scenario-based stochastic programs. In Marie Hušková, Petr Lachout, and Jan Ámos Víšek, editors, *Prague Stochastics '98*, pages 119–124. JČMF, 1998.
19. M. Ehrgott and D.M. Ryan. Constructing robust crew schedules with bicriteria optimization. *Journal of Multi-Criteria Decision Analysis*, 11(3):139–150, 2002.
20. Matthias Ehrgott and David M. Ryan. The method of elastic constraints for multiobjective combinatorial optimization and its application in airline crew scheduling. In T. Tanino, T. Tanaka, and M. Inuiguchi, editors, *Multi-Objective Programming and Goal Programming—Theory and Applications*, pages 117–122. Springer-Verlag, Berlin, 2003.
21. Y. Ermoliev. Stochastic quasigradient methods and their application to system optimization. *Stochastics*, 9:1–36, 1983.
22. J. Fairbrother, A. Turner, and S.W. Wallace. Scenario generation for single-period portfolio selection problems with tail risk measures: Coping with high dimensions and integer variables. *INFORMS Journal on Computing*, 30(3):472–491, 2018. https://doi.org/10.1287/ijoc.2017.0790.
23. J. Fairbrother, A. Turner, and S.W. Wallace. Problem-driven scenario generation: an analytical approach to stochastic programs with tail risk measure. *Mathematical Programming*, pages 141–182, 2022.
24. Yonghan Feng and Sarah M. Ryan. Solution sensitivity-based scenario reduction for stochastic unit commitment. *Computational Management Science*, pages 1–34, 2014. ISSN 1619-697X. https://doi.org/10.1007/s10287-014-0220-z.
25. Olga Fiedler and Werner Römisch. Stability in multistage stochastic programming. *Annals of Operations Research*, 56(1):79–93, 2005. https://doi.org/10.1007/BF02031701.
26. K. Froot and J. Stein. Risk management, capital budgeting and capital structure policy for financial institutions: An integrated approach. *Journal of Financial Economics*, 47:55–82, 1998.
27. A. Gaivoronski. Stochastic quasigradient methods and their implementation. In *Numerical techniques for stochastic optimization*, volume 10 of *Springer Ser. Comput. Math.*, pages 313–351. Springer, Berlin, 1988.
28. H. I. Gassmann and W. Ziemba. *Stochastic Programming: Applications in Finance, Energy, Planning and Logistics*. World Scientific Books. World Scientific Publishing Co. Pte. Ltd., 2013.
29. R.C. Grinold. Model building techniques for the correction of end effects in multistage convex programs. *Operations Research*, 31(4):407–431, 1983.
30. Zhaoxia Guo, Stein W. Wallace, and Michal Kaut. Vehicle routing with space- and time-correlated stochastic travel times: Evaluating the objective function. *INFORMS Journal on Computing*, 31(4):654–670, 2019. https://doi.org/10.1287/ijoc.2019.0906.
31. W.K. Klein Haneveld, M.H. van der Vlerk, and W. Romeijnders. *Stochastic Programming: Modeling Decision Problems Under Uncertainty*. Graduate Texts in Operations Research. Springer, 2020.

32. H. Heitsch and W. Römisch. Scenario reduction algorithms in stochastic programming. *Computational Optimization and Applications*, 24(2–3):187–206, 2003. https://doi.org/10.1023/A:1021805924152.
33. H. Heitsch and W. Römisch. Scenario tree reduction for multistage stochastic programs. *Computational Management Science*, 6(2):117–133, 2009. https://doi.org/10.1007/s10287-008-0087-y.
34. H. Heitsch, W. Römisch, and C. Strugarek. Stability of multistage stochastic programs. *SIAM Journal on Optimization*, 17(2):511–525, 2006. https://doi.org/10.1137/050632865.
35. Holger Heitsch and Werner Römisch. A note on scenario reduction for two-stage stochastic programs. *Operations Research Letters*, 35(6):731–738, 2007. https://doi.org/10.1016/j.orl.2006.12.008.
36. Réne Henrion and Werner Römisch. Problem-based optimal scenario generation and reduction in stochastic programming. *Mathematical Programming*, 191(1):183–205, 2022.
37. Mike Hewitt, Janosch Ortmann, and Walter Rei. Decision-based scenario clustering for decision-making under uncertainty. *Annals of Operations Research*, 315(2):747–771, Jan 2021. https://doi.org/10.1007/s10479-020-03843-x.
38. J. L. Higle and S. Sen. Stochastic decomposition: An algorithm for two-stage linear programs with recourse. *Mathematics of Operations Research*, 16:650–669, 1991.
39. J. L. Higle and S. Sen. Statistical verification of optimality conditions for stochastic programs with recourse. *Annals of Operations Research*, 30:215–240, 1991.
40. J. L. Higle and S. W. Wallace. Sensitivity analysis and uncertainty in linear programming. *Interfaces*, 33:53–60, 2003. https://doi.org/10.1287/inte.33.4.53.16370.
41. J.L. Higle. Variance reduction and objective function evaluation in stochastic linear programs. *INFORMS Journal on Computing*, 10(2):236–247, 1998.
42. K. Høyland and S. W. Wallace. Generating scenario trees for multistage decision problems. *Management Science*, 47(2):295–307, 2001. https://doi.org/10.1287/mnsc.47.2.295.9834.
43. Kjetil Høyland, Michal Kaut, and Stein W. Wallace. A heuristic for moment-matching scenario generation. *Computational Optimization and Applications*, 24(2–3):169–185, 2003. https://doi.org/10.1023/A:1021853807313.
44. Gerd Infanger. Monte Carlo (importance) sampling within a Benders decomposition algorithm for stochastic linear programs. *Ann. Oper. Res.*, 39(1–4):69–95 (1993), 1992. ISSN 0254-5330.
45. John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with AlphaFold. *Nature*, 596(7873):583–589, 2021.
46. P. Kall and J. Mayer. *Stochastic Linear Programming*. Springer, 2011.
47. P. Kall and S.W. Wallace. *Stochastic Programming*. Wiley, Chichester etc., 1994.
48. Michal Kaut. A copula-based heuristic for scenario generation. *Computational Management Science*, 11(4):503–516, 2014. https://doi.org/10.1007/s10287-013-0184-4.
49. Michal Kaut and Stein W. Wallace. Evaluation of scenario-generation methods for stochastic programming. *Pacific Journal of Optimization*, 3(2):257–271, 2007.
50. Michal Kaut and Stein W. Wallace. Shape-based scenario generation using copulas. *Computational Management Science*, 8(1–2):181–199, 2011. https://doi.org/10.1007/s10287-009-0110-y.
51. Michal Kaut, Stein W. Wallace, Hercules Vladimirou, and Stavros Zenios. Stability analysis of portfolio management with conditional value-at-risk. *Quantitative Finance*, 7(4):397–409, 2007. https://doi.org/10.1080/14697680701483222.
52. Julien Keutchayan, Janosch Ortmann, and Walter Rei. Problem-driven scenario clustering in stochastic optimization. *Computational Management Science*, 20(1), mar 2023. https://doi.org/10.1007/s10287-023-00446-2.
53. A. J. King. Asymmetric risk measures and tracking models for portfolio optimization under uncertainty. *Annals of Operations Research*, 45:165–177, 1993.
54. Alan King, Teemu Pennanen, and Matti Koivu. Calibrated option bounds. Technical report, Research Report RC22810, IBM Thomas J. Watson Research Center, 2003.

55. Alan J. King. Duality and martingales: A stochastic programming perspective on contingent claims. *Mathematical Programming*, 91(3):543–562, 2002.
56. Alan J. King, Olga Streltchenko, and Yelena Yesha. Private valuation of contingent claims: Discrete time/state model. In John Guerard, editor, *Handbook of Portfolio Construction: Contemporary Applications of Markowitz Techniques*, pages 691–710. Springer, 2009.
57. Anton J. Kleywegt, Alexander Shapiro, and Tito Homem-de Mello. The sample average approximation method for stochastic discrete optimization. *SIAM Journal on Optimization*, 12(2):479–502, 2001. https://doi.org/10.1137/S1052623499363220.
58. A. G. Kök, M.L. Fisher, and R. Vaidyanathan. Assortment planning: Review of literature and industry practice. In N. Agrawal and S.A. Smith, editors, *Retail Supply Chain Management*, pages 99–154. Springer Verlag, 2008.
59. Guanghui Lan and Zhiqiang Zhou. Dynamic stochastic approximation for multi-stage stochastic optimization. *Mathematical Programming*, 187:487–532, 2017.
60. A.-G. Lium, T. G. Crainic, and S. W. Wallace. A study of demand stochasticity in stochastic network design. *Transportation Science*, 43(2):144–157, 2009. https://doi.org/10.1287/trsc.1090.0265.
61. Leonard C. MacLean, Edward O. Thorp, and William T. Ziemba. *The Kelly capital growth investment criterion: theory and practice*. Handbook in Financial Economics. World Scientific Press, 2011.
62. S. Mahajan and G. van Ryzin. Retail inventories and consumer choice. In S. Tayur, R. Ganesham, and M. Magasine, editors, *Quantitative methods in Supply Chain Management*. Kluwer Publishers, Amsterdam, 1998.
63. W.K. Mak, D.P. Morton, and R.K. Wood. Monte Carlo bounding techniques for determining solution quality in stochastic programs. *Operations Research Letters*, 24:47–56, 1999.
64. H.M. Markowitz. *Portfolio selection: Efficient diversification of investment*. Yale University Press, New Haven, 1959.
65. Benjamin S. Narum, Jamie Fairbrother, and Stein W. Wallace. Problem-based scenario generation by decomposing output distributions. 2022.
66. Panos Parpas, Berk Ustun, Mort Webster, and Quang Kha Tran. Importance sampling in stochastic programming: A Markov chain Monte Carlo approach. *INFORMS Journal on Computing*, 27(2):358–377, 2015.
67. M.V.F. Pereira and L.M.V.G. Pinto. Multi-stage stochastic optimization applied to energy planning. *Mathematical Programming*, 52:359–375, 1991.
68. G. C. Pflug. Scenario tree generation for multiperiod financial optimization by optimal discretization. *Mathematical Programming*, 89(2):251–271, 2001. https://doi.org/10.1007/PL00011398.
69. G. Ch. Pflug and A. Pichler. *Multistage Stochastic Optimization*. Springer, 2014.
70. Georg Ch. Pflug and Alois Pichler. *Multistage Stochastic Optimization*. Springer Series in Operations Research and Financial Engineering. Springer, 2014. https://doi.org/10.1007/978-3-319-08843-3.
71. S Pineda and AJ Conejo. Scenario reduction for risk-averse electricity trading. *IET generation, transmission & distribution*, 4(6):694–705, 2010.
72. W.B. Powell. *Stochastic Delays in Transportation Terminals: New Results in the Theory and Application of Bulk Queues*. PhD thesis, Department of Civil Engineering, Massachusetts Institute of Technology, Cambridge, MA, U.S.A., 1981.
73. András Prékopa. *Stochastic programming*, volume 324 of *Mathematics and its Applications*. Kluwer Academic Publishers Group, Dordrecht, 1995. ISBN 0-7923-3482-5.
74. Vit Prochazka and Stein W. Wallace. Stochastic programs with binary distributions: structural properties of scenario trees and algorithms. *Computational Management Science*, 15:387–410, 2018.
75. Vit Prochazka and Stein W. Wallace. Scenario tree construction driven by heuristic solutions of the optimization problem. *Computational Management Science*, 17:277–307, 2020.
76. J. M. Rosenberger, E. L. Johnson, and G. L. Nemhauser. A robust fleet-assignment model with hub isolation and short cycles. *Transportation Science*, 38(3):357–368, 2004.

77. J.O. Royset and R.J-B Wets. *An Optimization Primer*. Springer International, 2021.
78. Andrej Ruszczynski and Alexanger Shapiro. *Stochastic Programming*. Handbooks in Operations Research and Management Science. Elsevier, 2002.
79. A. Shapiro. Monte Carlo sampling methods. In A. Ruszczyński and A. Shapiro, editors, *Stochastic Programming*, volume 10 of *Handbooks in Operations Research and Management Science*, chapter 6, pages 353–425. Elsevier Science B.V., Amsterdam, 2003. https://doi.org/10.1016/S0927-0507(03)10006-0.
80. Alexander Shapiro. Monte Carlo sampling approach to stochastic programming. *ESAIM: Proceedings*, 13:65–73, 2003. https://doi.org/10.1051/proc:2003003. Proceedings of 2003 MODE-SMAI Conference.
81. Gordon Sick. Real options. In *Finance*, volume 9 of *Handbooks in Operations Research and Management Science*, chapter 21, pages 631–691. Elsevier, 1995.
82. J.W. Suurballe and Robert E. Tarjan. A quick method for finding shortest pairs of paths. *Networks*, 14:325–336, 1984.
83. Hajnalka Vaagen and Stein W. Wallace. Product variety arising from hedging in the fashion supply chains. *International Journal of Production Economics*, 114(2):431–455, 2008. https://doi.org/10.1016/j.ijpe.2007.11.013.
84. J. von Neumann and O. Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, Princeton, New Jersey, 2nd edition, 1947.
85. Stein. W. Wallace. Decision making under uncertainty: Is sensitivity analysis of any use? *Operations Research*, 48:20–25, 2000. https://doi.org/10.1287/opre.48.1.20.12441.
86. Stein W. Wallace and William T. Ziemba, editors. *Applications of Stochastic Programming*. MPS-SIAM Series on Optimization. SIAM, Philadelphia, 2005.
87. Xin Wang, Teodor Gabriel Crainic, and Stein W. Wallace. Stochastic network design for planning scheduled transportation services: The value of deterministic solutions. *INFORMS Journal on Computing*, 31(1):153–170, 2019.
88. Wikipedia contributors. Physics-informed neural networks—Wikipedia, the free encyclopedia, 2022. https://en.wikipedia.org/wiki/Physics-informed_neural_networks. [Online; accessed 24-Feb-2023].
89. Wei Zhang, Kai Wang, Alexandre Jacquillat, and Shuaian Wang. Optimized scenario reduction: Solving large-scale stochastic programs with quality guarantees. *INFORMS Journal on Computing*, 2023.
90. Y. Zhao and S. W. Wallace. Integrated facility layout design and flow assignment problem under uncertainty. *INFORMS Journal on Computing*, 26(4):798–808, 2014.
91. W. T. Ziemba. personal communication, 2008.
92. Jikai Zou, Shabbir Ahmed, and Xu Andy Sun. Stochastic dual dynamic integer programming. *Mathematical Programming*, 175:461–502, 2019.

Index

A. J. King, S. W. Wallace, *Modeling with Stochastic Programming*, Springer Series in Operations Research and Financial Engineering,
https://doi.org/10.1007/978-3-031-54550-4