A primer on stochastic optimization for business applications

Paolo Brandimarte[0000−0002−6533−3055]

Abstract In this chapter we consider the application of optimization models under stochastic uncertainty, with emphasis on business applications like operations management and finance. Due to the strictly tutorial nature of the chapter, emphasis on model building, not on solution algorithms. We outline a range of problems from plain static decisions to multistage and adaptive models. Simple numerical examples, based on the classical newsvendor problem and decision trees, are used to help intuition building. Then, we illustrate stochastic programming models for production planning and asset–liability management. We also introduce relevant concepts like the expected value of perfect information (EVPI) and the value of the stochastic solution (VSS), and discuss issues related with scenario generation and stability.

1 Motivation: should we bother about uncertainty?

A good starting point to understand the nature of decision-making under uncertainty is a seemingly innocent question:

> Consider a real-valued random variable Y, for which we know the probability density function (PDF) $f_Y(y)$. Why do we use its expected value as a point forecast?

The problem should be framed as a stochastic optimization problem. First, we choose a real number, say x, as a point forecast. Then the random variable Y will be realized, and we will observe a forecast error $Y - x$. It is natural to consider the best point forecast as the one minimizing some *ex ante* measure, and a possible choice is mean squared error,

Paolo Brandimarte
Dipartimento di Scienze Matematiche, Politecnico di Torino, Corso Duca degli Abruzzi 24, 10129 Torino, Italy; e-mail: paolo.brandimarte@polito.it; URL: https://staff.polito.it/paolo.brandimarte

$$\mathsf{MSE}(x) \doteq \mathbb{E}\left[(Y-x)^2\right] = \mathbb{E}[Y^2] - 2x\mathbb{E}[Y] + x^2,$$

which is obviously minimized by choosing $x = \mathbb{E}[Y]$. However, is there anything wrong if we choose another measure? We could consider the absolute value of the error,

$$\mathbb{E}\left[\ |\ Y-x\ |\ \right],$$

in which case the optimal forecast is the median of Y, as we shall see later. Considering the absolute value implies that we penalize large errors a bit less than with a squared error, possibly resulting in less sensitivity to outliers. But actually, why should we consider *symmetric* error penalties? In a real life application, positive or negative errors may have different business impact. Hence, we may consider asymmetric penalties, like

$$\mathbb{E}\left[c_u(Y-x)^+ + c_o(Y-x)^-\ |\ \right],$$

where we define $z^+ \doteq \max\{0, z\}$ and $z^- \doteq \max\{0, -z\}$, and c_u and c_0 are underage and overage penalty coefficients, respectively. We will see a practical application of such penalties in Section 2.1, where we will prove that the optimal solution is a quantile of the distribution of Y, with a probability level depending on the overage and underage costs.

The essential message here is that a point forecast of uncertain risk factors, typically based on expected values, is not enough for decision making. We should consider the cost of being wrong in terms of *optimality*. But, what is the impact of being wrong in terms of *feasibility*? We will see how issues with feasibility are addressed in models with chance constraints or recourse variables.

Business decision models require the selection of a vector $\mathbf{x}$ of decision variables, which is constrained to lie within a feasible set $S \subset \mathbb{R}^n$. If the economic result is affected by a vector of uncertain risk factors $\boldsymbol{\xi} \in \mathbb{R}^l$, then we should consider an optimization problem under uncertainty like

$$\min_{\mathbf{x}\in S} f(\mathbf{x}, \boldsymbol{\xi}).$$

Clearly, this statement does not make mathematical sense and we must chose a precise model formulation. If we do not have any distributional knowledge about the uncertain risk factors, and all we know is that $\boldsymbol{\xi}$ belongs to an uncertainty set $\mathcal{U}$, we may formulate a worst-case robust optimization problem:

$$\min_{\mathbf{x}\in S}\left[\max_{\boldsymbol{\xi}\in\mathcal{U}} f(\mathbf{x}, \boldsymbol{\xi})\right].$$

If, on the contrary, we assume that the risk factors can be represented as a vector of random variables $\tilde{\boldsymbol{\xi}}$,[1] whose distribution is known and represented by a probability measure $\mathbb{P}$, then we could consider the stochastic programming problem

[1] When using Latin letters, we use capital letters like Y to refer to a random variable, and lowercase letters like y to refer to its realization. In the case of Greek letters, we use $\tilde{\xi}$ and ξ to refer to random variables and realizations, respectively.

$$\min_{\mathbf{x}\in S} \mathbb{E}_{\mathbb{P}}\left[f(\mathbf{x}, \tilde{\boldsymbol{\xi}})\right]. \tag{1}$$

In this chapter, we focus exclusively on stochastic optimization problems. It is fundamental to understand that we cannot disregard uncertainty and just rely on point forecasts, since, in general,

$$\mathbb{E}_{\mathbb{P}}\left[f(\mathbf{x}, \tilde{\boldsymbol{\xi}})\right] \neq f\left(\mathbf{x}, \mathbb{E}_{\mathbb{P}}[\tilde{\boldsymbol{\xi}}]\right).$$

This leads to the idea of stochastic optimization models, which come in a variety of shapes, as we discuss in this chapter.

2 Static decision models

The formulation of Eq. (1) suggests a static decision problem, whereby we make a decision $\mathbf{x}$ before the realization of the random variable $\tilde{\boldsymbol{\xi}}$, but there is no way to adapt the decision after the occurrence of the random event. This is best illustrated by a prototypical example of static optimization under uncertainty, the classical newsvendor model.

2.1 The classical newsvendor model

A simple example of static decision is the classical newsvendor problem, where we must decide the amount q of items to buy at unit cost c (the purchase cost), before observing the realization D of an uncertain demand. Each item is sold at a sales price $p > c$, but unsold items at the end of the day imply a loss. We may assume that there is some salvage value $r < c$ from each unsold item. An obvious question is what is the order quantity q that maximizes the expected value of profit,[2] $\mathbb{E}[\pi(q, D)]$. Should we just set $q = \mathbb{E}[D]$? A less obvious question is: Should we be conservative and reduce q when demand uncertainty increases? To see that the answer to the first question is negative, we consider a simple counterexample.

Example 1 Let us consider a discrete demand distribution, where values are integer and uniformly distributed between 5 and 15, so that the probability of each demand value is $1/11$ and $\mathbb{E}[D] = 10$. As to the economic data, the unit cost is $c = 20$, the sale price is $p = 25$, and unsold items have no salvage value ($r = 0$). Clearly, we must choose an integer value q between 5 and 15. For every choice of q, and a given scenario where the realized demand is $D = d$, profit is given by

[2] For the sake of simplicity, we assume risk-neutrality. Hence, we only care about expected profit, without considering its variability.

Table 1 Expected profit for the newsvendor problem of Example 1.

q	5	6	7	8	9	10
$\mathbb{E}[\pi(q,D)]$	25.00	27.73	28.18	26.36	22.27	15.91
q	11	12	13	14	15	
$\mathbb{E}[\pi(q,D)]$	7.27	-3.64	-16.82	-32.27	-50.00	

$$\pi(q,d) = \begin{cases} (p-c)q & \text{if } q \le d \\ pd - cq & \text{if } q > d. \end{cases}$$

Let us consider a couple of possible choices. If we are very conservative and set $q = 5$, there will never be any leftover item, and we always sell 5 items in every scenario. Hence, profit is deterministic and given by $(25-20)\times 5 = 25$. If we choose $q = 6$, profit will be $(25-20)\times 6 = 30$ in each scenario where $D \ge 6$, whereas we have $(25-20)\times 5 - 20 = 5$ if $D = 5$. Therefore,

$$\mathbb{E}[\pi(6,D)] = \frac{5 + 10\times 30}{11} = 27.73,$$

which is a better value. Going on this way, we obtain the values listed in Table 1. Clearly, $q = \mathbb{E}[D] = 10$ is not the optimal choice, and the optimal solution is given by a lower value, $q^* = 7$.

We should wonder why the optimal solution in Example 1 is smaller than expected demand. Arguably, this may be explained by the unfavorable problem economics: the profit margin is much smaller than the loss on unsold items. To see this, we need a better solution approach than number crunching. For given values of q and d, a general expression of profit is

$$\pi(q,d) = -cq + p\min(q,d) + r\max(q-d,0) = -cq + p\min(q,d) + r(q-d)^+,$$

where $(x)^+ \doteq \max(x,0)$. Given the identity

$$q \doteq \min(q,d) + (q-d)^+$$

we may rewrite profit as

$$\pi(q,d) = c_u \min(q,d) - c_o(q-d)^+ \tag{2}$$

where we define the cost of underage $c_u \doteq p - c$ and the cost of overage $c_o \doteq c - r$. If demand uncertainty is modeled by a continuous random variable with density $f_D(x)$, expected profit is

$$\begin{aligned}\mathbb{E}\left[\pi(q,D)\right] =& c_u \left(\int_0^q x f_D(x)\, dx + \int_q^{+\infty} q f_D(x)\, dx \right) \\ & - c_o \int_0^q (q-x) f_D(x)\, dx. \end{aligned} \tag{3}$$

To find its maximum, let us recall the Leibniz integral rule to take the derivative of a function defined as

$$G(q) = \int_{h_1(q)}^{h_2(q)} g\,(q,x)\; dx.$$

Under suitable conditions, we have :

$$\frac{dG}{dq}(q) = \int_{h_1(q)}^{h_2(q)} \frac{\partial g}{\partial q}\,(q,x)\, dx + g\big(q, h_2\,(q)\,\big) \cdot h_2'\,(q) - g\big(q, h_1\,(q)\,\big) \cdot h_1'\,(q)\,.$$

Applying this result to the first-order optimality condition on the expected profit, we obtain

$$\begin{aligned}\frac{d\mathbb{E}[\pi(q,D)]}{dq} &= c_u \left(q \cdot f_D(q) + \int_q^{+\infty} f_D(x)\, dx - q \cdot f_D(q) \right) - c_o \int_0^q f_D(x)\, dx \\ &= c_u \int_q^{+\infty} f_D(x)\, dx - c_o \int_0^q f_D(x)\, dx \\ &= c_u \left(1 - F_D(q)\right) - c_o F_D(q) = 0,\end{aligned}$$

where $F_D(x) \doteq \mathbb{P}\{D \leq x\}$ is the (cumulative) distribution function of demand. Therefore, we find that the optimal quantity q^* satisfies the condition

$$F\,(q^*) = \frac{c_u}{c_u + c_o}. \tag{4}$$

To prove that the first-order condition is sufficient, we may also check the concavity of the objective function with respect to q:

$$\frac{d^2\mathbb{E}\left[\pi(q,D)\right]}{dq^2} = -c_u\, f_D(q) - c_o\, f_D(q) < 0, \qquad \forall q.$$

The condition (4) is easy to interpret: the optimal order quantity is a quantile of the demand distribution, depending on problem economics. Indeed, q is large when c_u is large with respect to c_o. When the two penalty coefficients are the same, the optimal solution is the median of the demand distribution, as hinted at in Section 1.

The newsvendor model is clearly a static one, as we make a single decision with no possibility of adaptation. We will consider a more flexible version of the newsvendor model in Section 6.2. Another reason why the newsvendor problem is easy is that there is no feasibility concern. What if we have to meet an uncertain inequality constraint $g(\mathbf{x}, \tilde{\boldsymbol{\xi}}) \leq 0$? Chance constraints, which we discuss next, are a natural way to state reliability requirements in a single-stage, static model.

2.2 Chance-constrained models

The idea of a chance constraint is that we are satisfied when a constraint like $g(\mathbf{x}, \tilde{\boldsymbol{\xi}}) \leq 0$ is satisfied with a suitably large probability. We may consider individual chance constraints, like[3]

$$\mathbb{P}\{g_j(\mathbf{x}, \tilde{\boldsymbol{\xi}}) \leq 0\} \geq 1 - \alpha_j, \qquad j \in [m], \tag{5}$$

or we may collect these constraints into a joint chance constraint,

$$\mathbb{P}\{\mathbf{g}(\mathbf{x}, \tilde{\boldsymbol{\xi}}) \leq \mathbf{0}_m\} \geq 1 - \alpha, \tag{6}$$

where $\mathbf{g}$ is a vector-valued function. Intuition would suggest that when functions g_j are convex with respect to $\mathbf{x}$, for evey $\boldsymbol{\xi}$, this should translate to convexity of the probabilistic constraint. Unfortunately, this is not the case in general.

Example 2 (A non-convex chance-constrained problem)
Let us consider the capacity constraints of a simple optimal production mix problem:

$$\begin{aligned} 2x_1 + x_2 &\leq R_1 \\ x_1 + 2x_2 &\leq R_2, \end{aligned}$$

where $x_1, x_2 \geq 0$ represent the amount produced of items 1 and 2, and R_1 and R_2 are the available capacity (hours) of two resources. In the nominal scenario, which happens with probability $\pi_a = 0.9$, we have $R_1 = R_2 = 300$. In this case, the set of feasible production plans is the polytope with North-East vertex $\mathbf{x}_a = (100, 100)$, as shown in Fig. 1. However, say that there are two alternative scenarios, both occurring with probability $\pi_b = \pi_c = 0.05$, where capacity is reduced due a fault on resource 1 or 2. In scenario b, the North-East vertex becomes $\mathbf{x}_b = 1/3(100, 400)$, whereas in scenario c we have $\mathbf{x}_c = 1/3(400, 100)$. Now, suppose that we want a production plan that stays feasible with probability 0.94. Since this reliability is less than both $\pi_a + \pi_b = 0.95$ and $\pi_a + \pi_c = 0.95$, it is easy to see that the feasible set is the *union* of the two smaller polyhedra, which is not a convex set.

Chance-constrained models may be inconvenient, when they lack convexity properties. In fact, robust optimization has been proposed as a way to devise safe convex approximations to non-convex chance-constrained problems. Nevertheless, they may be a useful tool for engineering design problems, where adaptation is impossible. However, in a business problem, we often can take advantage of the sequential flow of

[3] We use the shorthand notation $[m] \doteq \{1, \ldots, m\}$.

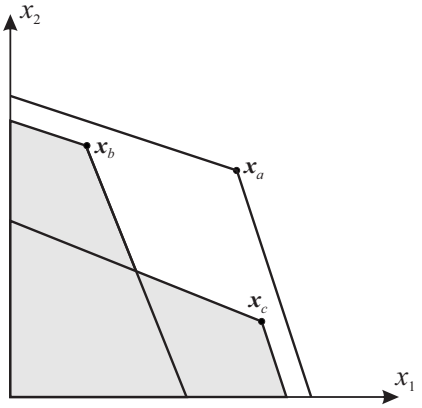

Fig. 1 A non-convex feasible region associated with a chance constraint.

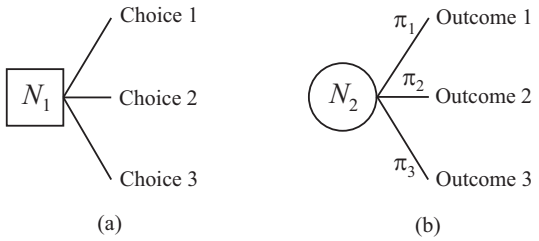

Fig. 2 Node types in a decision tree: (a) decision nodes, where choices are made; (b) chance nodes, where random outcomes are selected by "nature" according to probabilities.

information, leading to adaptive, multistage models. As a first step towards adaptive stochastic optimization, let us consider next simple decision trees.

3 From static to adaptive decision models: Decision trees

A good way to introduce adaptive decisions under uncertainty is by considering decision trees. A decision tree comprises two essential kinds of node:

- *Decision nodes*, represented by squares, correspond to discrete choices between mutually exclusive alternatives, as depicted in Fig. 2(a). At these nodes, the decision maker must choose one among multiple available options.
- *Chance nodes*, represented by circles, correspond to the realization of random outcomes, as depicted in Fig. 2(b). Each outcome i is associated with a probability π_i; clearly, the probabilities of the random outcomes at each chance node add up to 1.

A decision tree consists of a set of decision and chance nodes, as shown in Fig. 3. Typically, decision and chance nodes are interleaved. We also have *terminal nodes*,

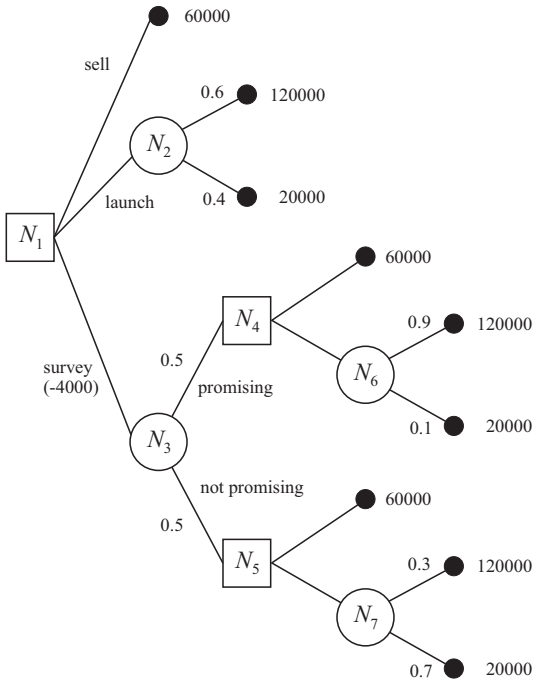

Fig. 3 A sample decision tree.

represented by bullets, which are typically associated with monetary outcomes, as shown in Figure 3. Solving the problem means choosing a *strategy*, i.e., selecting, for each decision node that we *might* visit, one among multiple alternative decisions. Assuming that payoffs have a monetary nature, the most natural criterion to follow in building the strategy is the maximization of the *expected monetary value* (EMV). When a decision node is followed by a set of chance nodes, we should label each chance node with an EMV, which allows us to choose the best action at the decision node. The labeling process should go backward in time, starting from terminal nodes, as illustrated by the following example.[4]

Example 3 (New product introduction) The tree of Figure 3 represents the following decision problem. A firm has developed a new product and it must decide whether starting full scale production is worthwhile. If the product is successful, profit will be €120,000; otherwise, it will just be €20,000. Probability of success is 0.6, or 60%, so there is quite some uncertainty. As an alternative, the firm could just sell the production license for €60,000 to another firm. However, there is still a third possibility; the firm could try to get some additional information by carrying out a market research survey, at a

[4] This example is adapted from the text by Curwin and Slater [9].

cost of €4,000. If the result of the survey is promising, the firm believes that the conditional probability of success will be 0.9; otherwise, it will just be 0.3. Note that these are *conditional* probabilities, given the outcome of the survey.

Before investigating the best course of action for the firm, let us observe that, apparently, there is a missing piece of information: the probability p that the outcome from the survey is promising (an event that we denote as $\mathsf{S}_{\mathsf{good}}$). However, we should realize that running the survey does not imply changing the product, but only gathering information. Hence, it should not change the unconditional probability of success, which is 60% if we do not carry out the survey. If we carry out the survey, the unconditional probability of success, as seen from the root of the tree, is

$$\begin{aligned}\mathbb{P}(\mathsf{P}_{\mathsf{OK}}) &= \mathbb{P}(\mathsf{P}_{\mathsf{OK}} \,|\, \mathsf{S}_{\mathsf{good}}) \times \mathbb{P}(\mathsf{S}_{\mathsf{good}}) + \mathbb{P}(\mathsf{P}_{\mathsf{OK}} \,|\, \mathsf{S}_{\mathsf{bad}}) \times \mathbb{P}(\mathsf{S}_{\mathsf{bad}}) \\ &= 0.9 \times p + 0.3 \times (1 - p)\end{aligned}$$

Setting this equal to 0.6 and solving for the unknown probability yields $p = 0.5$. This is the only probability ensuring consistency in our representation of uncertainty. The expected profit from immediate product launch is

$$\mathbb{E}[\pi] = 0.6 \times 120{,}000 + 0.4 \times 20{,}000 = 80{,}000, \tag{7}$$

a value that we use to label the chance node N_2. If we care only about expected values, i.e., if we are risk-neutral, this is larger than the profit from selling a production license.

If we carry out the survey, and the result is promising, the conditional expected profit is

$$\mathbb{E}\left[\pi \,|\, \mathsf{S}_{\mathsf{good}}\right] = 0.9 \times 120{,}000 + 0.1 \times 20{,}000 = 110{,}000$$

from which the cost of the survey should be deducted. If the survey is not promising, then

$$\mathbb{E}\left[\pi \,|\, \mathsf{S}_{\mathsf{bad}}\right] = 0.3 \times 120{,}000 + 0.7 \times 20{,}000 = 50{,}000$$

and we are better off selling the license. Hence, if we carry out the survey and make optimal use of the information that it provides, we obtain

$$\mathbb{E}[\pi] = 0.5 \times 110{,}000 + 0.5 \times 60{,}000 - 4000 = 81{,}000 \tag{8}$$

We see that the best course of action is:

- Carrying out the survey.
- If the result is promising, start production.
- If the result is not promising, sell the license.

We may also observe that the effect of the survey is to reduce uncertainty, and the firm should be willing to pay at most €5000 for this reduction. To see this,

observe that the expected profit in Eq. (8), without deducting the cost of the survey, is €85,000, which should be compared against €80,000, the expected profit of immediate product launch from Eq. (7).

The example shows that decision trees may provide an estimate of the value of information. For the tree of Fig. 3 we know that the value of the *partial* information provided by the customers' survey is not larger than €5000. In the next section we show how to value *perfect* information.

4 EVPI and VSS

Consider the optimal value of a static stochastic optimization problem

$$f^* = \min_{\mathbf{x} \in S} \mathbb{E}_{\mathbb{P}} \left[f(\mathbf{x}, \tilde{\boldsymbol{\xi}}) \right]. \tag{9}$$

How would this value improve, if we could postpone the here-and-now decision and wait and see which scenario is realized? Formally, we should swap minimization and expectation in Eq. (9):

$$f^*_{\text{PI}} = \mathbb{E}_{\mathbb{P}} \left[\min_{\mathbf{x} \in S} f(\mathbf{x}, \tilde{\boldsymbol{\xi}}) \right]. \tag{10}$$

The subscript in f^*_{PI} tells us that this is the expected value of cost if we could optimize under perfect information. It stands to reason, and it can be shown formally, that for a minimization problem

$$f^*_{\text{PI}} \leq f^*$$

The difference in cost is the **expected value of perfect information**

$$\text{EVPI} \doteq f^* - f^*_{\text{PI}} \tag{11}$$

The EVPI tells us something about the impact of uncertainty and is best illustrated by a toy example.

Example 4 Consider a stylized investment problem. We must select an investment strategy between two possibilities, aggressive and conservative. These two choices are associated with decisions x_1 and x_2, respectively; so, the feasible set is $S = \{x_1, x_2\}$. Uncertainty is represented by three possible states of the economy, which we interpret as follows:

- Scenario ω_1, with probability 0.1, is a "very bullish" economy, in which the aggressive strategy yields 30% and the conservative one yields a less exciting +6%.

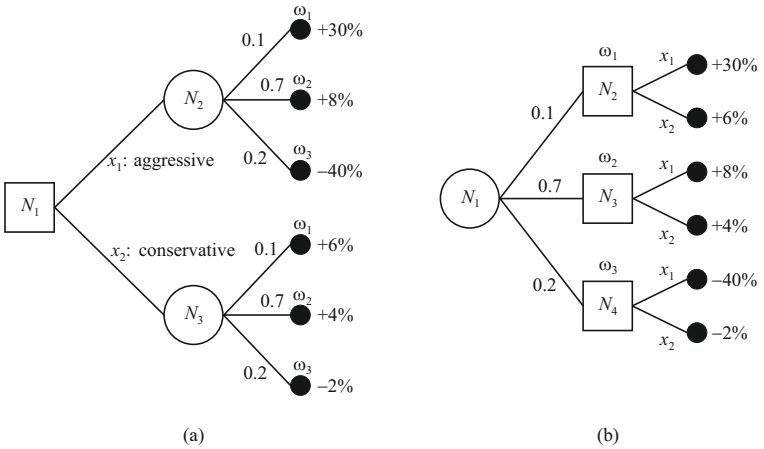

Fig. 4 Schematic illustration of EVPI – the value of (waiting for) perfect information: (a) here-and-now and (b) wait-and-see decisions.

- Scenario ω_2, with probability 0.7, is a "moderately bullish" economy, in which the aggressive strategy yields 8% and the conservative one yields +4%.
- Scenario ω_3, with probability 0.2, is a "bearish" economy, in which the aggressive strategy loses an epic 40% and the conservative one limits the loss to a moderate 2%.

Here we are using $\omega \in \Omega$ to denote the outcome within a sample space Ω. The random variable $\tilde{\boldsymbol{\xi}}$ is a mapping from outcomes (more generally, events) to numerical realizations $\boldsymbol{\xi}(\omega)$. The here-and-now decision consists of selecting a strategy before observing returns. The corresponding decision tree is depicted in Fig. 4(a). If we take expected return as our objective function, to be maximized, we should take the maximum between

$$f_1 = 0.1 \times 30\% + 0.7 \times 8\% + 0.2 \times (-40\%) = 6\%$$

and

$$f_2 = 0.1 \times 6\% + 0.7 \times 4\% + 0.2 \times (-2\%) = 3\%$$

So, we would choose the aggressive strategy, and $f^* = 6\%$. Now, how much would be the value of clairvoyance? We should restructure the decision tree as in Fig. 4(b). If we could invest after observing return, the expected return before the realization of the state of the economy would be

$$f^*_{\text{PI}} = 0.1 \times \max\{30\%, 6\%\} + 0.7 \times \max\{8\%, 4\%\} + 0.2 \times \max\{-40\%, -2\%\} = 8.2\%$$

The EVPI, in terms of percentage returns, is

$$\text{EVPI} = f^*_{\text{PI}} - f^* = 8.2\% - 6\% = 2.2\%.$$

Of course, we swap terms with respect to Definition (11), as we seek to maximize return. The return could be translated into monetary terms by assuming an invested wealth.

In financial markets, the impact of clairvoyance would be actually larger than suggested by the example, as we could leverage the investment by borrowing cash to invest in the best strategy and further increase the return. Clearly, one has quite rarely access to perfect information. Example 3 shows a more limited impact of partial information in a different context. In fact, EVPI is in most cases a theoretical construct, and a possibly more practical concept can be obtained by pursuing a different line of reasoning.

We cannot swap minimization and expectation in problem (9), but we could entertain the idea of making our life a lot simpler by disregarding uncertainty, and replace the random risk factors by their expected value. By doing so, we would solve deterministic "expected value" problem

$$f^*_{\text{EV}} = \min_{\mathbf{x}\in S} f(\mathbf{x}, \mathbb{E}_{\mathbb{P}}[\tilde{\boldsymbol{\xi}}]),$$

which yields the "expected-value solution" $\bar{\mathbf{x}}$. This model also yields a value of the objective function, but the solution $\bar{\mathbf{x}}$ must be evaluated within the actual uncertain setting. The actual cost of fixing decision $\bar{\mathbf{x}}$ and then observing the actual realization $\tilde{\boldsymbol{\xi}}$ leads to a random cost $f(\bar{\mathbf{x}}, \tilde{\boldsymbol{\xi}})$. If we take its expectation, we define the expected value of the expected-value solution:

$$f_{\text{EEV}} \doteq \mathbb{E}_{\mathbb{P}}[f(\bar{\mathbf{x}}, \tilde{\boldsymbol{\xi}})].$$

What we should compare is f_{EEV} against f^*, whereas f^*_{EV} is misleading. The **value of the stochastic solution** (VSS) is defined as

$$\text{VSS} \doteq f_{\text{EEV}} - f^*$$

for a minimization problem (swapping terms is needed for a maximization problem). It can be shown that VSS is nonnegative [3]. When VSS is large, the additional effort in solving the much more complicated stochastic optimization problem does pay off.

We illustrate VSS by a numerical example in the next section, which is also an introduction to two-stage optimization models.

5 An introductory two-stage model: Assemble-to-order

Let us consider a toy example of production planning, where end items are obtained by assembling a set of components. Each end item is characterized by a set of

Table 2 Bill of materials for the assemble-to-order example.

	c_1	c_2	c_3	c_4	c_5
A_1	1	1	1	0	0
A_2	1	1	0	1	0
A_3	1	1	0	0	1

Table 3 Bill of resources, cost of components, and available capacity (Cap.).

	M_1	M_2	M_3	Cost
c_1	1	2	1	20
c_2	1	2	2	30
c_3	2	2	0	10
c_4	1	2	0	10
c_5	3	2	0	10
Cap.	800	700	600	

features, for which alternative *options* are available, requiring different components. Note that a huge range of end items may result from combining a limited set of components. Hence, we may not hedge against demand uncertainty by keeping end items in stock. On the other hand, a pure make-to-order strategy may be discouraged by a possibly long delivery lead time for components. In an Assemble-To-Order (ATO) environment, where assembly is fast, we may keep components in stock, and using them to assemble end items after we observe demand. Clearly, this suggest a two-stage strategy:

1. we plan production of components *here and now*, under uncertainty about the demand for end items;
2. we assemble end items in a *wait and see* fashion, after receiving customer orders.

To be concrete, let us consider a toy numerical example. Say that we produce three types of end item (A_1, A_2, A_3), which are obtained by assembling five types of component (c_1, c_2, c_3, c_4, c_5). The components that we need for each end item are described by a bill of materials, which is flat (just two levels: end items and components). The bill of materials is given in Table 2. For instance, to assemble a single end item of type A_2, we need one component of type c_1, one of type c_2, and one of type c_4. From the bill of materials, we see that there are two common components, c_1 and c_2, while the remaining three are specific and characterize each end item. We assume that three resource types (machine groups M_1, M_2, and M_3) are used for the production of components. Table 3 lists:

- The bill of resources, i.e., the time required on each resource to produce one component of each type.
- The available capacity (time available) for each resource class.
- The cost of each component, which might include both direct variable production costs and material costs.

Since we assume that assembly is not a bottleneck, we do not list any resource consumption or capacity information concerning assembly. We note that the cost of every end item type is $20 + 30 + 10 = 60$; we do not make assembly cost explicit, but that would be easy to include.

Now, we also need data on the market side, which are listed in Table 4. The last column gives the price at which end items are sold. For all of the three end items,

Table 4 Demand scenarios, expected demand, and sale prices of end items.

	s_1	s_2	s_3	exp.demand	sale price
A_1	100	50	120	90	80
A_2	50	25	60	45	70
A_3	100	110	60	90	90

the selling price is larger than 60, the total component cost; however, A_3 looks more profitable, because its contribution to profit is $90 - 60 = 30$, whereas A_2 is the least profitable. Demand uncertainty is modeled by three equally likely scenarios s_1, s_2, and s_3. In this case, the expected value of demand is just an average, as reported in Table 4.

Let us assume that, in order to keep life simple, we ignore demand uncertainty and just consider its average. We introduce following the decision variables:

- $x_i \in \mathbb{Z}_+$, $i = 1, \ldots, 5$, the integer number of produced components;
- $y_j \in \mathbb{Z}_+$, $j = 1, \ldots, 3$, the integer number of assembled end items.

To maximize profit, we need to solve an integer linear program:

$$\max \quad -\sum_{i\in[5]} C_i x_i + \sum_{j\in[3]} P_j y_j \tag{12}$$

$$\text{s.t.} \quad \sum_{i\in[5]} T_{im} x_i \le L_m, \qquad m \in [3] \tag{13}$$

$$y_j \le \overline{d}_j, \qquad j \in [3] \tag{14}$$

$$\sum_{j\in[3]} G_{ij} y_j \le x_i, \qquad i \in [5] \tag{15}$$

$$x_i, y_j \in \mathbb{Z}_+ \equiv \{0, 1, 2, 3, \ldots\}.$$

Here, we denote the time required to process component i on machine group m by T_{im}, and the time available for each group by L_m. The expected demand for each end item j is denoted by $\overline{d}_j$, and G_{ij} is the number of component i needed to assemble end item j. By using any commercial solver, we obtain:

$$x_1^* = 116, \quad x_2^* = 116, \quad x_3^* = 26, \quad x_4^* = 0, \quad x_5^* = 90,$$
$$y_1^* = 26, \quad y_2^* = 0, \quad y_3^* = 90,$$

with an objective value $f_{\text{EV}} = 3220$.

In this very small example, we may easily interpret what this solution is trying to accomplish. We assemble the maximum number of end items of type A_3, subject to its demand limitation $d_3 = 90$, since this is the most profitable one; this requires in turn the production of a corresponding number of common components c_1 and c_2,

and of specific component c_5. Since the market limitation is binding for A_3, there is some capacity left, which is used to produce a limited amount of the specific component c_3, which is needed to assemble end item A_1, plus the corresponding number of common components. End item A_2 has the lowest selling price and is disregarded, as is its specific component c_4. It should be noted that, in general, one should not take for granted that the production of the highest profit item must be maximized; the consumption of available resources should be taken into account as well. The numerical solution of this toy example is very easy to interpret, but it should also be taken with utmost care. Any experienced production planner would be very critical about its "extreme" nature: It is essentially a bet on demand of the most profitable item. This might make sense if we are quite sure about demand, but what if actual demand turns out to be rather different?

In order to account for uncertainty, we need:

- to represent uncertainty by scenario-dependent demand d_j^s, $s \in \{1, 2, 3\}$, with probability π^s;
- to introduce a wait-and-see (second-stage) decision variable $y_j^s \in \mathbb{Z}_+$, allowing for adaptation of assembly decisions to observed orders.

We obtain a two-stage stochastic linear programming model:

$$\max \quad -\sum_{i\in[5]} C_i x_i + \sum_{s\in[3]} \pi^s \left(\sum_{j\in[3]} P_j y_j^s \right) \tag{16}$$

$$\text{s.t.} \quad \sum_{i\in[5]} T_{im} x_i \le L_m, \qquad m \in [3] \tag{17}$$

$$y_j^s \le d_j^s, \qquad j \in [3], s \in [3] \tag{18}$$

$$\sum_{j\in[3]} G_{ij} y_j^s \le x_i, \qquad i \in [5], s \in [3] \tag{19}$$

$$x_i, y_j^s \in \mathbb{Z}_+.$$

By solving the stochastic optimization model, we obtain

$$\begin{aligned}
&x_1^* = 115, \quad x_2^* = 115, \quad x_3^* = 55, \quad x_4^* = 0, \quad x_5^* = 65,\\
&y_1^{1*} = 50, \quad y_2^{1*} = 0, \quad y_3^{1*} = 65,\\
&y_1^{2*} = 50, \quad y_2^{2*} = 0, \quad y_3^{2*} = 65,\\
&y_1^{3*} = 55, \quad y_2^{3*} = 0, \quad y_3^{3*} = 60,
\end{aligned}$$

with expected profit $f^* = 2883.33$. The actual output of the model only consists of the here-and-now decisions, as the wait-and-see assembly decisions are just contingency plans to make the solution more robust. In real life, we would observe a demand scenario that need not be part of the planned scenarios, and we would solve an optimization model to plan assembly, given the available components. We clearly

see that this solution is less extreme than the previous one, and that the amounts of common components is not really different, due to a *risk pooling* effect.

We could be puzzled by the fact that the resulting expected profit, $f^* = 2883.33$, is less than the optimal profit from the expected value problem, $f_{\text{EV}} = 3220$. As we have pointed out when discussing VSS, this comparison does not make any sense, as in the deterministic model we *pretend* to know end item demand and we get the illusion of higher profits. In order to compare the two solutions, we should fix the production plans for components that the two models propose, and then we should solve a set of second-stage problems, where we optimize assembly of end items subject to component availability, for different demand scenarios. More formally, given the vector $\mathbf{x}^\circ$ of first-stage optimal decisions of whatever model, we should solve the following second-stage problem for each scenario s in $\mathcal{S} = \{1, 2, 3\}$:

$$\begin{aligned} R^s(\mathbf{x}\circ) \doteq \max \quad & \sum_{j\in[3]} p_j y_j^s \\ \text{s.t.} \quad & y_j^s \le d_j^s, \qquad j \in [3] \\ & \sum_{j\in[3]} g_{ij} y_j^s \le x_i^\circ, \qquad i \in [5] \\ & y_j^s \in \mathbb{Z}_+. \end{aligned}$$

where $R^s(\mathbf{x}^\circ)$ is the optimal revenue that we collect under scenario s, given the first-stage solution $\mathbf{x}^\circ$, by making optimal use of the available components to meet demand. Note that in this model the component availability x_i° is given, either by the stochastic or by the deterministic model. Whatever the case, the resulting expected revenue is

$$\sum_{s\in\mathcal{S}} \pi^s R^s(\mathbf{x}^\circ).$$

In the three scenarios in the example, we find the following assembly plans:

$$\begin{aligned} y_1^1 = 26, \quad y_2^1 = 0, \quad y_3^1 = 90, \\ y_1^2 = 26, \quad y_2^2 = 0, \quad y_3^2 = 90, \\ y_1^3 = 26, \quad y_2^3 = 0, \quad y_3^3 = 60. \end{aligned}$$

We clearly see that scenario s_3 would prove a disaster for the deterministic solution. In that scenario, sales are lower for A_3, but we could not react, because we do not hold enough specific components for the other end items. So, 30 specific components c_5 would be scrapped. Furthermore, common components would be scrapped as well, since they cannot be used to assemble other end items for the lack of the related specific components. As a consequence, the actual expected profit from the expected value solution is $f_{\text{EEV}} = 2320$, much less than the misleading and optimistic value $f_{\text{EV}} = 3220$. In our toy example, we find

$$\mathsf{VSS} = f^* - f_{\text{EEV}} = 2883.33 - 2320 = 563.33.$$

6 Two-stage models

The assemble-to-order model of the previous section is a good introduction to two-stage stochastic linear programming models with recourse, which are the simplest example of adaptive optimization models. In a two-stage model, we consider two types of decisions:

1. the *here-and-now* (first stage) decisions $\mathbf{x}$, which have to be made under uncertainty (say, at time $t = 0$);
2. the *wait-and-see* (second stage) decisions $\mathbf{y}(\tilde{\boldsymbol{\xi}})$, which can be made after observing the realization of risk factors $\tilde{\boldsymbol{\xi}}$ (say, at time $t = T$).

Second stage decisions are also called recourse decisions. We immediately see that such decisions are actually decision *policies*, i.e., functions mapping the realization of risk factors into a suitable decision. As such, they live in an infinite-dimensional space.

We may formulate the model as two nested optimization problems:

$$\min_{\mathbf{x}} \quad \mathbf{c}^{\mathsf{T}}\mathbf{x} + Q(\mathbf{x}) \tag{20}$$

$$\text{s.t.} \quad \mathbf{A}\mathbf{x} = \mathbf{b} \tag{21}$$

$$\mathbf{x} \geq \mathbf{0},$$

where we define **recourse function**

$$Q(\mathbf{x}) \doteq \mathbb{E}_{\mathbb{P}}\left[Q(\mathbf{x}, \tilde{\boldsymbol{\xi}})\right], \tag{22}$$

and the second-stage problem

$$Q(\mathbf{x}, \boldsymbol{\xi}) \doteq \min_{\mathbf{y}} \quad \mathbf{q}(\boldsymbol{\xi})^{\mathsf{T}}\mathbf{y} \tag{23}$$

$$\text{s.t.} \quad \mathbf{W}\mathbf{y} = \mathbf{h}(\boldsymbol{\xi}) - \mathbf{T}(\boldsymbol{\xi})\mathbf{x} \tag{24}$$

$$\mathbf{y} \geq \mathbf{0}.$$

In the second-stage problem, both the first-stage decision $\mathbf{x}$ and the realization $\boldsymbol{\xi}$ of the random risk factors are given. By solving the second-stage problem for every realization $\boldsymbol{\xi}$ of the random variable $\tilde{\boldsymbol{\xi}}$, we implicitly define a function $\mathbf{y}(\tilde{\boldsymbol{\xi}})$, showing that the second-stage variables are actually functions. Here, the recourse matrix $\mathbf{W}$ does not depend on random variables, and we speak of **fixed recourse**. This formulation shows that stochastic linear programming with recourse is, in general, a nonlinear programming problem.

The recourse function $Q(\mathbf{x})$ is an expectation with respect to the joint distribution of $\tilde{\boldsymbol{\xi}}$; hence, it is a multidimensional integral, if random variables are continuous. Moreover, it is a multidimensional integral of a function that we do not really know, as it is implicitly defined by an optimization problem. Luckily, in many cases of practical interest, we can prove interesting properties of the recourse function, most notably convexity.

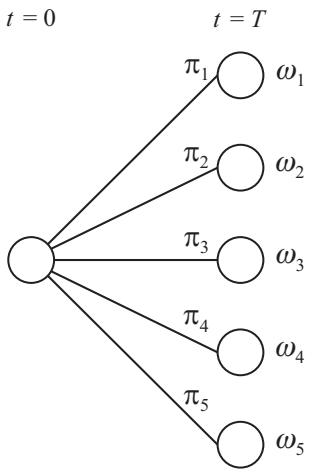

Fig. 5 A scenario tree for a two-stage model.

Since second-stage variables are actually functions living in an infinite-dimensional space, a common solution strategy relies on discretization by sampling. We generate a scenario tree (fan), like the one illustrated in Fig. 5, where ω_s is the event corresponding to the realization of scenario $s \in \mathcal{S}$, where $\mathcal{S}$ is the set of scenarios. If scenarios are sampled by plain Monte Carlo, then scenario probabilities are uniform: $\pi_s = 1/|\mathcal{S}|$. More sophisticated scenario generation methods may be used. This yields the following model:

$$\begin{aligned} \min \quad & \mathbf{c}^\top \mathbf{x} + \sum_{s \in \mathcal{S}} \pi_s \mathbf{q}_s^\top \mathbf{y}_s \\ \text{s.t.} \quad & \mathbf{A}\mathbf{x} = \mathbf{b} \\ & \mathbf{W}\mathbf{y}_s + \mathbf{T}_s \mathbf{x} = \mathbf{h}_s, \qquad s \in \mathcal{S} \\ & \mathbf{x}, \mathbf{y}_s \geq \mathbf{0}. \end{aligned} \tag{25}$$

This is a plain LP, even though a possibly large-scale one.

A relevant question for two-stage models is whether the second-stage model is feasible for whatever choice of the first-stage variables and every realization of the random variables. We should restrict the feasible set of here-and-now decisions to a domain where the second-stage problem is feasible (i.e., the recourse function is bounded above; as usual, we associate an infinite cost to an infeasible problem). We may rearrange the linking constraints (25) as

$$\mathbf{W}\mathbf{y}_s = \mathbf{h}_s - \mathbf{T}_s \mathbf{x}, \qquad s \in \mathcal{S},$$

and observe that the second-stage problem is feasible whenever we may express the right-hand side vector as a conic combination of the columns of the recourse matrix $\mathbf{W}$. If this is the case, we speak of **complete recourse**. Sometimes, complete recourse

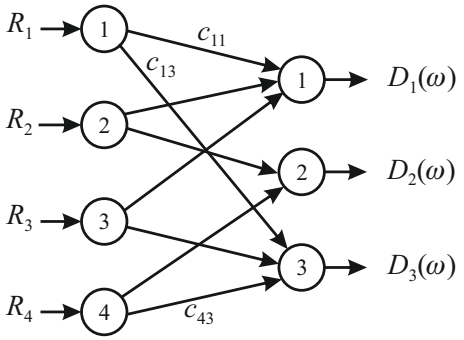

Fig. 6 Bipartite network corresponding to a plant location model.

may not apply for arbitrary first-stage decisions, but it does hold for first-stage decisions satisfying the first-stage constraints. In this case, we speak of **relatively complete recourse**.

Usually, in a business problem, we have some degree of flexibility that helps in formulating the model by the introduction of suitable penalties, making sure that the first-stage solution does not induce infeasibilities in the second stage. In the next two sections we illustrate the idea by concrete examples.

6.1 The plant location model

The classical plant location model is the simplest network design problem. We have a bipartite network, as shown in Fig. 6, consisting of a set $\mathcal{P}$ of potential source nodes and a set $\mathcal{D}$ of demand nodes. Source nodes are locations of production plants that we may build or not. Depending on the application, demand nodes may be regional warehouses or retail centers. At demand nodes, a random demand $D_j(\omega)$, $j \in \mathcal{D}$, is realized, which we should satisfy at minimum cost. To formulate the model we need the following data:

- for each $i \in \mathcal{P}$ we have a fixed opening cost f_i and a capacity level R_i;
- for each $j \in \mathcal{D}$ and scenario $s \in \mathcal{S}$, we have a demand d_j^s (we assume a single product type for simplicity); this is a suitable discretization of the distribution of random demand $D_j(\omega)$;
- for each arc (i, j) in the network we have a unit transportation cost c_{ij} (assuming linear variable costs).

It is important to observe that the problem can be naturally cast as a two-stage problem, as we have to build the network under demand uncertainty, but transportation decisions may be delayed until we observe actual demand. Hence, let us define the decision variables:

$$y_i = \begin{cases} 1 & \text{if source node } i \in \mathcal{P} \text{ is opened} \\ 0 & \text{otherwise} \end{cases}$$

and $x_{ij}^s \geq 0$, the amount of flow from source node $i \in \mathcal{P}$ to destination node $j \in \mathcal{D}$ in scenario $s \in \mathcal{S}$. The binary location variables are first-stage, *design* variables. On the contrary, the transportation variables are second-stage, *control* variables.

Let us consider the following two-stage mixed-integer linear programming (MILP) model with recourse:

$$\begin{aligned} \min \quad & \sum_{i \in \mathcal{P}} f_i y_i + \sum_{s \in \mathcal{S}} \pi^s \left(\sum_{i \in \mathcal{P}} \sum_{j \in \mathcal{D}} c_{ij} x_{ij}^s \right), \\ \text{s.t.} \quad & \sum_{i \in \mathcal{P}} x_{ij}^s = d_j^s, \qquad \forall s \in \mathcal{S},\ \forall j \in \mathcal{D}, \\ & \sum_{j \in \mathcal{D}} x_{ij}^s \leq R_i y_i, \qquad \forall s \in \mathcal{S},\ \forall i \in \mathcal{P}, \\ & x_{ij}^s \geq 0,\ y_i \in \{0,1\}. \end{aligned}$$

Now we should pause a little, and wonder whether there is anything wrong with this model formulation. What if an extreme scenario with a huge realization of demand is included? On the one hand, we may not be sure that such a demand realization can be satisfied at all. Even so, it is clear that the capacity plan will be driven by extreme, but possibly very unlikely demand scenarios, resulting in a very expensive solution. Sometimes, we should resort to *elastic* model formulations, by allowing for suitably penalized constraint violations. Let $z_j^s \geq 0$ be the amount of unmet demand at node j under scenario s. These decision variables should be included in the objective function multiplied by a penalty coefficient β_j. We may formulate the model as:

$$\begin{aligned} \min \quad & \sum_{i \in \mathcal{P}} f_i y_i + \sum_{s \in \mathcal{S}} \pi^s \left(\sum_{i \in \mathcal{P}} \sum_{j \in \mathcal{D}} c_{ij} x_{ij}^s \right) + \sum_{s \in \mathcal{S}} \pi^s \left(\sum_{j \in \mathcal{D}} \beta_j z_j^s \right), \\ \text{s.t.} \quad & \sum_{i \in \mathcal{P}} x_{ij}^s + z_j^s = d_j^s \qquad \forall s \in \mathcal{S},\ \forall j \in \mathcal{D}, \\ & \sum_{j \in \mathcal{D}} x_{ij}^s \leq R_i y_i \qquad \forall s \in \mathcal{S},\ \forall i \in \mathcal{P}, \\ & x_{ij}^s, z_j^s \geq 0,\ y_i \in \{0,1\}. \end{aligned}$$

The quantification of the penalty coefficients β_j depends on the exact nature of the slack variables z_j^s. They may literally represent unmet demand, in which case we may wish to differentiate the penalties to reflect the relative priorities of different markets. However, they may represent demand that is satisfied by resorting to emergency production by a third-party supplier, in which case they should represent the actual cost of doing so.

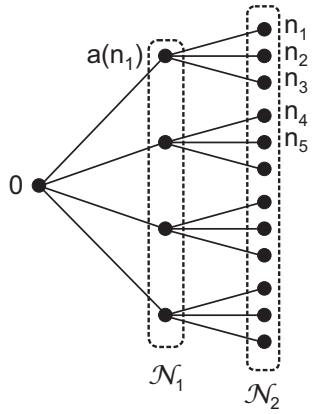

Fig. 7 Scenario tree for a two-stage newsvendor problem.

The model that we have considered is clearly just a first step towards a more realistic model, in which transportation costs may be nonlinear, and multiple item types are considered. Moreover, we are disregarding time and the potential use of inventories to hedge against partially predictable demand variability over time. This would lead to a much more demanding multistage model. The model in this section relies on a simple *anticipation function* to approximate the expectation about future transportation cost. The relevant output is just the selected network design.

6.2 A two-stage newsvendor model

Let us consider a two-stage, multi-item version of the newsvendor problem, subject to capacity constraints. The model is motivated by a sport fashion producer, who has to start production early, under severe demand uncertainty, due to capacity constraints. However, it may adjust demand forecasts and production plans after observing a first portion of sales to retailers. See [16] for an illustration of the real life business case. A formal model is discussed in [10].

The information structure may be represented by the three-stage scenario tree of Fig. 7. At node 0 we have to make a here-and-now decision on the first production lot sizes for each item (SKU – Stock-Keeping Unit). The set $\mathcal{N}_1$ of intermediate nodes corresponds to states in which we have observed early sales and have to make decisions on the second set of production lots, under reduced uncertainty. Finally, at terminal (leaves) nodes in set $\mathcal{N}_2$ we observe actual sales and check overage and underage costs, exactly like in the newsvendor model. Each node $n \in \mathcal{N}_2$ has exactly one predecessor (antecedent) node $a(n) \in \mathcal{N}_1$. The root node 0 is the direct antecedent of each node $n \in \mathcal{N}_1$

It is important to realize that the resulting model looks like a three-stage model, but the last decisions are fictitious ones, as they represent surplus or a shortfall

amounts, which are penalized by overage and underage penalties, respectively. They are neither design nor control variables; they are just accounting variables. Unlike the basic newsvendor model, here we have capacity constraints and limits on the minimum lot sizes.

To formalize the model, we need the following sets and data:

- Let $\mathcal{I}$ be the set of items (SKUs).
- Let K_1 and K_2 denote the maximum number of items that we can produce in the first and the second production run, respectively. We assume that the processing times for different SKUs are similar, so that we may express capacity in terms on how many items we may produce.
- We also have minimum lot sizes m_i, $i \in \mathcal{I}$. Note that the lot sizes are *semicontinuous* variables: a minimum lot size does not mean that there is a minimum amount to be produced. The idea is that we may not produce an item, but if we do produce it, there is a minumum production run, below which production is not economically justified (due, e.g., to setup costs).
- Demand for each SKU $i \in \mathcal{I}$ and scenario $n \in \mathcal{N}_2$, associated with probability $\pi^{(n)}$, is denoted by d_i^n.
- Finally, for each SKU $i \in \mathcal{I}$, we have an overage (surplus) penalty coefficient c_i^o and an underage (shortfall) penalty c_i^u.

Then, we introduce the decision variables:

- The amount produced for each node $n \in \{0\} \cup \mathcal{N}_1$, denoted by x_i^n, which is a non-negative integer number.
- For each for each production node $n \in \{0\} \cup \mathcal{N}_1$, we also introduce binary variables $\delta_i^n \in \{0, 1\}$, set to 1 if we produce item i at that node, 0 otherwise. These variables are needed to represent semicontinuous variables.
- Finally, the accounting variables u_i^n and o_i^n represent the deviations, shortfall/underage or surplus/overage, with respect to the demand observed for each item $i \in \mathcal{I}$ at each leaf node $n \in \mathcal{N}_2$.

Finally, here is the model formulation, which is a stochastic MILP:

$$\min \quad \sum_{n \in \mathcal{N}_2} \pi^n \left[\sum_{i \in \mathcal{I}} \left(c_i^o o_i^n + c_i^u u_i^n \right) \right] \tag{26}$$

$$\text{s.t.} \quad \sum_{i \in \mathcal{I}} x_i^0 \leq K_1 \tag{27}$$

$$x_i^0 \geq m_i \delta_i^0, \quad x_i^0 \leq K_1 \delta_i^0 \qquad i \in \mathcal{I} \tag{28}$$

$$\sum_{i \in \mathcal{I}} x_i^n \leq K_2 \qquad n \in \mathcal{N}_1 \tag{29}$$

$$x_i^n \geq m_i \delta_i^n, \quad x_i^n \leq K_2 \delta_i^n \qquad i \in \mathcal{I},\ n \in \mathcal{N}_1 \tag{30}$$

$$x_i^0 + x_i^{a(n)} = d_i^n + o_i^n - u_i^n \qquad i \in \mathcal{I},\ n \in \mathcal{N}_2 \tag{31}$$

$$x_i^n \in \mathbb{Z}_+,\ \delta_i^n \in \{0, 1\} \qquad i \in \mathcal{I},\ n \in \{0\} \cup \mathcal{N}_1$$

$$u_i^n,\ o_i^n \geq 0 \qquad i \in \mathcal{I},\ n \in \mathcal{N}_2.$$

The objective function (26) is the expected deviation cost. Equations (27) and (29) are the capacity constraints (maximum number of items produced), for the two production runs. Equations (28) and (30) are the classical big-M constraints to link continuous and binary decision variables; in this case, we use production capacity as the big-M. Note that when the binary variable is 1, production is constrained to lie between the minimum lot size and capacity; when the binary variable is set to 0, the lot size is squeezed and forced to 0. Finally, Eq. (31) measures the deviation between the total produced amount and actual demand. Note that we may avoid enforcing an integrality restriction on the deviation variables, as this is implied by the integrality restrictions on the production variables.

6.3 Multistage stochastic linear programming with recourse

Multistage stochastic programming formulations arise naturally as a generalization of two-stage models. Conceptually, we just have to nest recourse functions corresponding to decision stages. We may apply the formalism of Section 6, but in this case we should consider that future decisions might depend on the whole history of the stochastic process of risk factors. Let us introduce the notation

$$\tilde{\boldsymbol{\xi}}_{[t]} \doteq (\tilde{\boldsymbol{\xi}}_1, \tilde{\boldsymbol{\xi}}_2, \ldots, \tilde{\boldsymbol{\xi}}_t),$$

where, clearly, $\tilde{\boldsymbol{\xi}}_{[1]} \equiv \tilde{\boldsymbol{\xi}}_1$. We still have to make a here-and-now decision $\mathbf{x}_0 \in X_0$, by solving a problem

$$\min_{\mathbf{x}_0 \in X_0} \left\{ f_0(\mathbf{x}_0) + \mathbb{E}\left[Q_1(\mathbf{x}_0, \tilde{\boldsymbol{\xi}}_1) \right] \right\}.$$

However, now $Q_1(\mathbf{x}_0, \boldsymbol{\xi}_1)$ is recursively defined as

$$Q_1(\mathbf{x}_0, \boldsymbol{\xi}_1) = \min_{\mathbf{x}_1 \in X_1(\mathbf{x}_0, \boldsymbol{\xi}_1)} \left\{ f_1(\mathbf{x}_1, \boldsymbol{\xi}_1) + \mathbb{E}\left[Q_2\left(\mathbf{x}_1, \tilde{\boldsymbol{\xi}}_{[2]}\right) \mid \tilde{\boldsymbol{\xi}}_1 = \boldsymbol{\xi}_1 \right] \right\}.$$

By the same token, for $t = 2, \ldots, T$, we have

$$Q_t(\mathbf{x}_{t-1}, \boldsymbol{\xi}_{[t]}) = \min_{\mathbf{x}_t \in X_t(\mathbf{x}_{t-1}, \boldsymbol{\xi}[t])} \left\{ f_t(\mathbf{x}_t, \boldsymbol{\xi}_{[t]}) + \mathbb{E}\left[Q_{t+1}\left(\mathbf{x}_t, \tilde{\boldsymbol{\xi}}_{[t+1]}\right) \mid \tilde{\boldsymbol{\xi}}_{[t]} = \boldsymbol{\xi}_{[t]} \right] \right\}. \tag{32}$$

We may define the recourse function

$$Q_{t+1}\left(\mathbf{x}_t, \boldsymbol{\xi}_{[t]}\right) \doteq \mathbb{E}\left[Q_{t+1}\left(\mathbf{x}_t, \tilde{\boldsymbol{\xi}}_{[t+1]}\right) \mid \tilde{\boldsymbol{\xi}}_{[t]} = \boldsymbol{\xi}_{[t]} \right]$$

and rewrite (32) as

$$Q_t(\mathbf{x}_{t-1}, \boldsymbol{\xi}_{[t]}) = \min_{\mathbf{x}_t \in X_t(\mathbf{x}_{t-1}, \boldsymbol{\xi}[t])} \left\{ f_t(\mathbf{x}_t, \boldsymbol{\xi}_{[t]}) + Q_{t+1}\left(\mathbf{x}_t, \boldsymbol{\xi}_{[t]}\right) \right\}. \tag{33}$$

In the the last stage problem, for $t = T$, we may just have $Q_T\left(\mathbf{x}_T, \boldsymbol{\xi}_{[T]}\right) \equiv 0$.

An alternative, possibly clearer formulation is obtained by emphasizing that recourse decisions are actually (non-anticipative) functions of the sample path observed so far, $\mathbf{x}_t\left(\tilde{\boldsymbol{\xi}}_{[t]}\right)$. This leads to the functional formulation

$$\begin{aligned}\min_{\mathbf{x}_0, \mathbf{x}_1(\cdot), \ldots, \mathbf{x}_T(\cdot)} \quad & \mathbb{E}\Big[f_0(\mathbf{x}_0) + f_1\Big(\mathbf{x}_1\left(\tilde{\boldsymbol{\xi}}_{[1]}\right), \tilde{\boldsymbol{\xi}}_1\Big) + \cdots + f_T\Big(\mathbf{x}_T\left(\tilde{\boldsymbol{\xi}}_{[T]}\right), \tilde{\boldsymbol{\xi}}_T\Big)\Big] \\ \text{s.t.} \quad & \mathbf{x}_0 \in \mathcal{X}_0 \\ & \mathbf{x}_t\left(\boldsymbol{\xi}_{[t]}, \boldsymbol{\xi}_T\right) \in \mathcal{X}_t\Big(\mathbf{x}_{t-1}\left(\boldsymbol{\xi}_{[t-1]}, \boldsymbol{\xi}_{t-1}\right)\Big), \qquad t = 1, \ldots, T.\end{aligned}$$

This formulation looks clearer, but the choice between the two models depends on the kind of solution strategy. The nested formulation lends itself to a recursive decomposition strategy, whereas the functional formulation suggests constraining the policy functions to lie in a finite-dimensional subspace of simple policies. However, since advanced solution strategies are beyond the scope of this tutorial (see the references in Section 9), it is best to consider a simple approach based on scenario trees, like the one depicted in Fig. 8.

A multistage scenario tree is a generalization of the two-stage fan of Fig. 5. Each scenario ω_k corresponds to a sequence of nodes, i.e., to a sample path

$$\boldsymbol{\xi}_{[T]}(\omega_k) = \big(\boldsymbol{\xi}_1(\omega_k), \boldsymbol{\xi}_2(\omega_k), \ldots, \boldsymbol{\xi}_T(\omega_k)\big).$$

The tree may be regarded as a discretization of the underlying stochastic process. A set of decision variables is associated with each node. Hence, a vector of variables $\mathbf{x}^n$ is associated with each node $n \in \mathcal{N}$, where $\mathcal{N}$ is the set of nodes in the tree. Therefore, rather than a function living in an infinite-dimensional space, we are dealing with a finite-dimensional problem.

For each time layer, we have a set of nodes. Each node is associated with exactly one time instant (so, we can eliminate time indexes), and each node n has exactly one antecedent $a(n)$, with the exception of the root node n_0, which has no antecendent node. On the other hand, each node has a set of direct successor nodes, and we have conditional probabilities $\pi_{k|j}$, which give the conditional probability of visiting node n_k after visiting the antecedent node n_j.

The tree structure is fundamental to represent non-anticipativity of decisions. Consider, for instance, scenarios

$$\omega_2 = \big(n_0, n_1, n_3, n_8\big) \qquad \text{and} \qquad \omega_3 = \big(n_0, n_1, n_4, n_9\big).$$

These two scenarios are not distinguishable at time instants $t = 0$ and $t = 1$, where they both visit nodes n_0 and n_1. The corresponding decision variables $\mathbf{x}^{n_0}$ and $\mathbf{x}^{n_1}$ cannot exploit future knowledge about the two scenarios. Only at time $t = 2$ we gather further information, allowing us to distinguish between these two scenarios. Obviously, at the root node we have to make a single, here-and-now decision, common to all of the scenarios. Only at later times we can exploit additional, bust still partial, information. This is best illustrated by a concrete example.

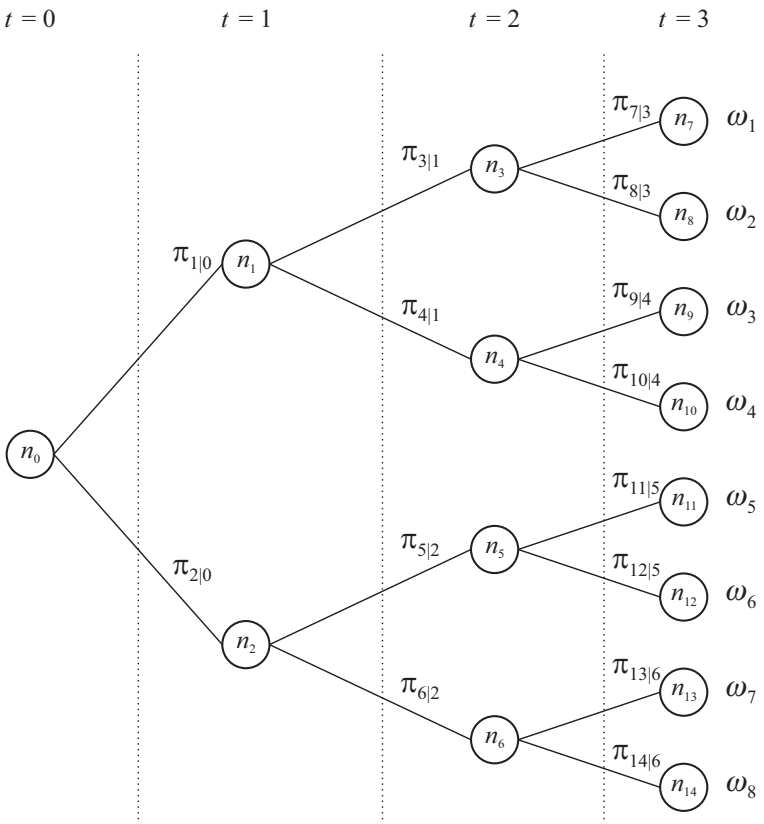

Fig. 8 A multistage scenario tree.

6.4 A multistage financial planning model with proportional transaction costs

We formulate an asset–liability management model, where we have to trade a set of financial instruments (the assets) in order to meet a stream of liabilities (e.g., pension payments for a pension fund).[5] We represent uncertainty by a scenario tree.

- We consider a stream of stochastic liabilities, where L^n is the liability to be met at each node $n \in \mathcal{N}$.
- For the sake of simplicity, we do not consider the possibility of receiving new cash deposits along the way, as it would be the case, e.g., for a pension fund receiving new contributions. The only way to raise cash is by selling assets.
- Assets are indexed by $i \in \mathcal{I}$, and we denote by P_i^n the price of asset $i \in \mathcal{I}$ at node $n \in \mathcal{N}$.

[5] For an overview of optimization models in finance, including stochastic and robust optimization models, see [8].

- We consider a simple form of transaction cost. Whenever we buy or sell assets, we incur proportional (linear) transaction costs; the transaction cost is a percentage c of the traded value, for both buying and selling any asset.[6]

The root node is n_0, where we have to make a first asset allocation decision, considering the initial holdings $\overline{h}_i^{n_0}$ for each asset $i \in \mathcal{I}$. The set of terminal nodes (leaves) is $\mathcal{S}$, where we may wish to consider the terminal value of wealth. The asset portfolio may be further rebalanced at trading nodes in the set

$$\mathcal{T} = \mathcal{N} \setminus (\{n_0\} \cup \mathcal{S}),$$

where $\setminus$ denotes set difference. We want to maximize the expected utility of terminal wealth at the leaves of the tree.

To account for transaction costs, we need decision variables measuring the number of units that we hold of each asset, e.g., number of bonds or stock shares. We must also specify how many units of each asset we buy or sell. Thus, we introduce the following decision variables:

- z_i^n, the amount of asset i purchased at node n;
- y_i^n, the amount of asset i sold at node n;
- x_i^n, the amount of asset i held at node n, after rebalancing;
- W^s, the wealth at terminal node $s \in \mathcal{S}$.

Variables z_i^n, y_i^n, and x_i^n are only defined at nodes $n \in \mathcal{N} \setminus \mathcal{S}$, as we assume that no rebalancing occurs at terminal nodes, where we liquidate the portfolio, pay the last liability L^s, and measure the utility of terminal wealth W^s. Let $u(W)$ be the nonlinear utility of wealth W.

On the basis of this notation, we may write the following model:

$$\max \quad \sum_{s \in \mathcal{S}} \pi^s u(W^s) \tag{34}$$

$$\text{s.t.} \quad x_i^{n_0} = \overline{h}_i^{n_0} + z_i^{n_0} - y_i^{n_0}, \qquad \forall i \in \mathcal{I} \tag{35}$$

$$x_i^n = x_i^{a(n)} + z_i^n - y_i^n, \qquad \forall i \in \mathcal{I},\ \forall n \in \mathcal{T} \tag{36}$$

$$(1-c) \sum_{i \in \mathcal{I}} P_i^n y_i^n - (1+c) \sum_{i \in \mathcal{I}} P_i^n z_i^n = L^n, \qquad \forall n \in \mathcal{N} \setminus \mathcal{S} \tag{37}$$

$$W^s = (1-c) \sum_{i \in \mathcal{I}} P_i^s x_i^{a(s)} - L^s, \qquad \forall s \in \mathcal{S} \tag{38}$$

$$x_i^n, z_i^n, y_i^n, W^s \geq 0. \tag{39}$$

The objective (34) is the expected utility of terminal wealth. Equation (35) expresses the initial asset balance, taking the current holdings into account. The asset balance at intermediate trading dates, i.e., at all nodes with the exception of root and leaves, is

[6] It is easy to generalize the model by introducing proportional transaction costs depending on the liquidity of each single asset, as well as the sign of the trade.

taken into account by Eq. (36). Equation (37) ensures that enough cash is generated by selling assets in order to meet the liabilities; we may also reinvest the proceeds of what we sell in new asset holdings. Note how proportional transaction costs are expressed for selling and purchasing. Effectively, we sell at a price that is lower than the price at which we buy. Equation (38) is used to evaluate terminal wealth at leaf nodes, after portfolio liquidation and payment of the last liability. In practice, we would repeatedly solve the model on a rolling-horizon basis, so the exact expression of the objective function is a bit debatable. The role of terminal utility is just to ensure that we are left in a good position at the end of the planning horizon, in order to avoid nasty end-of-horizon effects. In real life, the model should be extended to account for possible contributions, as well as suitably penalized shortfalls with respect to liabilities.

7 Multiperiod vs. multistage models

A clear issue with multistage stochastic programming models is the potential explosion of the scenario tree. The number of nodes needed to account for uncertainty may be a severe limiting factor.[7] Such an issue is less relevant for a tree structure like the ine depicted in Fig. 9.

It is important to realize that such a linear tree structure does *not* correspond to a multistage model, but rather to a two-stage, multiperiod model. Indeed, multiperiod does *not* mean multistage. We may even have a multiperiod but static model, if every decision is made here-and-now, but implemented at later time instants, without any adaptation to the flow of information. On the contrary, multistage implies adaptation, subject to non-anticipativity of the decision policies. The tree of Fig. 9 is somewhat peculiar, as it is two-stage, but it involves multiple time periods. Apparently, it violates the non-anticipativity condition, since after we observe the first branching after the root node, we may perfectly forecast the future sample path. Nevertheless, such a structure may make sense when we have to make design or structural decisions here-and-now, which cannot be adapted later, followed by a stream of control decisions where we cannot take advantage of the linear tree structure. The energy planning model of the next section illustrates such a case.

7.1 A two-stage multiperiod model: Unit commitment

Let us consider a stochastic model for a unit commitment problem, an energy planning problem where uncertainty in demand is modeled by a discrete-time stochastic process:

[7] We discuss remedies and approaches to scenario generation in Section 8. An additional issue with financial optimization models is related with the generation of spurious arbitrage opportunities. See, e.g., [4] and [13].

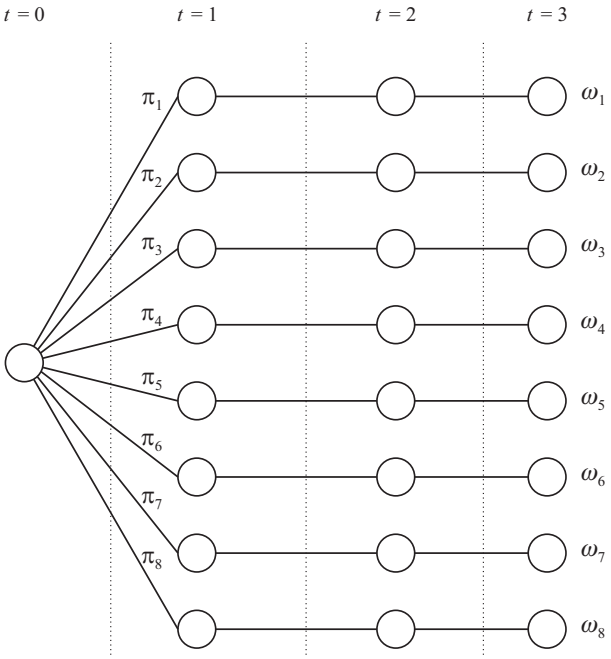

Fig. 9 A linear scenario tree for a multiperiod two-stage problem

$$\boldsymbol{\xi}(\omega) = \big(d_1(\omega), \ldots, d_T(\omega)\big), \qquad t \in [T],\ \omega \in \Omega,$$

where T is the length of the planning horizon and Ω represents the set of sample paths, which we assume finite. Let π^ω be the probability of scenario ω.

We have a set of I classes of generating units, which may provide an output in the range $[m_i, M_i]$, $i \in [I]$, when active. We have a_i units available for each class i. Again, the power output is a semicontinuous decision variable, which may also be left to zero, but has a minimum level m_i for an active unit. If a generating unit of class i is active at time t, we incur a cost E_i for keeping it at the minimum level m_i, and a linear variable cost C_i for each additional unit of power delivered. We also incur a startup cost s_i for switching a unit of class i on.[8]

We introduce a non-negative integer decision variable u_{it} denoting the total number of units of class i active during time interval t. Given the cost structure, there is no need to represent each single unit. We also consider a non-negative integer variable s_{it} representing the number of units of class i that are switched on at the beginning of time interval t. These are first-stage decision variables. On the contrary, the non-negative variable q_{it}^ω, representing the total output of class i during time interval t is scenario-indexed, second-stage decision variable.

[8] For the sake of illustration, we do not consider costs associated with switching a generator off, as well as possible ramp-up constraints, and minimum time that a state on/off must be conserved after a switch-on/off (a nervous pattern would damage the power unit).

The model can be formulated as:

$$\min \quad \sum_{i\in[I],t\in[T]} \left[E_i u_{it} + F_i s_{it}\right] + \sum_{\omega\in\Omega} \pi^\omega \left[\sum_{i\in[I],t\in[T]} C_i(q_{it}^\omega - m_i u_{it})\right] \tag{40}$$

$$\text{s.t.} \quad \sum_{i\in[I]} q_{it}^\omega \geq d_t(\omega), \qquad t \in [T],\ \omega \in \Omega \tag{41}$$

$$m_i u_{it} \leq q_{it}^\omega \leq M_i u_{it}, \qquad i \in [I], t \in [T], \omega \in \Omega \tag{42}$$

$$s_{it} \geq n_{it} - n_{i,t-1}, \qquad i \in [I], t \in [T] \tag{43}$$

$$u_{it} \leq a_i, \qquad i \in [I], t \in [T] \tag{44}$$

$$u_{it} \in \mathbb{Z}_+,\ s_{it} \in \mathbb{Z}_+,\ q_{it}^\omega \geq 0, \quad i \in [I], t \in [T], \omega \in \Omega.$$

The objective function (40) is self-explanatory. The only point worth noting is that the cost coefficient C_i must multiply only the additional power delivered above the minimum level m_i, multiplied by the number of active units. Constraints (41) ensure satisfaction of demand for each time period. Constraints (42) keep the total power delivered by class i during time period t between the minimum and the maximum levels. Constraints (43) set the correct values for the startup variables (note that s_{it} is zero when we switch generators off or when we keep the same number as the previous time period). Finally, (44) bounds the number of active units by the number available.

There are a couple of important remarks to be made about this model.

1. The first one concerns feasibility. If we insist on making the demand satisfaction constraint (41) a hard one, to be satisfied no matter what, we may end up with a very expensive plan, essentially driven by a handful of unlikely scenarios featuring huge demand spikes. Hence, just like we did with the plant location model, we should make the model elastic, maybe by including in a 'last-resort' generating unit, with unbounded capacity, playing the role of an external energy provider, or a penalty for unmet demand.
2. The second one concerns the flow of information. The model is multiperiod, but two-stage, and after the first demand is observed at time $t = 1$, we pretend that we know all of the demand values for periods $t \in [2 : T]$. This is not realistic, and we could wonder whether we should resort to a multistage model, where non-anticipativity requirements must be explicitly enforced. However, in this case, there is no way to take advantage of this information, since the design variables are here-and-now. Furthermore, there is no state in terms of energy, as energy cannot be stored (according to the model). Hence, in terms of power delivered, all of the time periods are unrelated.

8 Scenario generation and stability in stochastic programming

The key ingredient in multistage stochastic programming models is a scenario tree, and the quality of the solution depends critically on how well uncertainty is captured. In principle, we might just sample a dynamic model of risk factors using Monte Carlo simulation. Unfortunately, there are additional complications.

- The sample size must be kept limited, because we are going to associate decision variables with states, and the computational burden of solving a stochastic optimization problem, rather than taking a sample mean, may be remarkable.
- We need to generate non-anticipative decisions, which requires a tree structure prone to an exponential increase of complexity. If we branch 100 successor nodes out of each node in the tree, we have one million scenarios after three steps.
- We are not only evaluating an expected value, like we do in plain Monte Carlo simulation, but we are making decisions by solving an optimization problem. hence, any inconsistency or bias in the scenario tree will be exploited by the optimization solver, leading to a flawed solution, which looks good in-sample, but is likely to perform poorly out-of-sample.

Plenty of research effort has been dedicated to scenario generation, which we can only hint at. Whatever scenario generation strategy we select, it is important to check the stability of the resulting solution, as we briefly discuss in Section 8.1.

There are two basic classes of scenario generation approaches.

1. *Stochastic scenario generation.* Stochastic scenario generation is based on random Monte Carlo sampling, which relies on the brute force of a large sample size. This is something that we may afford in a two-stage problem, not so much in a multistage setting. However, performance may be sometimes improved by applying variance reduction strategies, such as importance sampling.
2. *Deterministic scenario generation.* Rather than using brute force, we may try to come up with a clever choice of scenarios, using an array of methods including Gaussian quadrature, low-discrepancy sequences (like Sobol sequences), or optimized scenario generation.

All of these strategies may be interpreted within the general framework of numerical integration, also called numerical quadrature. To see why, let us consider the expected value of a function $f(\mathbf{x}, \tilde{\boldsymbol{\xi}})$ depending on a vector $\mathbf{x}$ of decision variables and a vector of random variables $\tilde{\boldsymbol{\xi}}$ with joint density $h(\boldsymbol{\xi})$. By the integration with respect to $\boldsymbol{\xi}$, we define a function $F(\mathbf{x}) \doteq \mathbb{E}\left[f(\mathbf{x}, \tilde{\boldsymbol{\xi}})\right]$, which may be approximated by a random sample of size S as follows:

$$F(\mathbf{x}) = \int f(\mathbf{x}, \boldsymbol{\xi}) h(\boldsymbol{\xi})\, d\boldsymbol{\xi} \approx \frac{1}{S} \sum_{s=1}^{S} f(\mathbf{x}, \boldsymbol{\xi}_s),$$

where $\boldsymbol{\xi}_s$ is an observation in the sample. However, we may use a clever (possibly deterministic) sample of points $\boldsymbol{\xi}_s$, associated with weights π_s, and use the approximation

$$F(\mathbf{x}) \approx \sum_{s=1}^{S'} \pi_s f\left(\mathbf{x}, \boldsymbol{\xi}_s\right),$$

where $S' < S$. In the context of stochastic optimization, the pairs $(\pi_s, \boldsymbol{\xi}_s)$ may be understood in terms of probabilities and values, i.e., as a discrete distribution approximating the original continuous distribution. Gaussian quadrature formulas provide us with ways to select the discretization, and low-discrepancy sequences are used in quasi-Monte Carlo simulation, where the sample is generated in order to guarantee a "good" coverage of the integration domain.[9]

Example 5 (Scenario generation by moment matching) Consider a random variable with a multivariate normal distribution, $\tilde{\boldsymbol{\xi}} \sim \mathsf{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. The expected values and covariances of the discrete approximation should match those of the continuous distribution. Furthermore, since we are dealing with a normal distribution, we know that skewness and kurtosis for each individual variable should be 0 and 3, respectively.

Let us denote by ξ_i^s the realization of the component ξ_i of $\boldsymbol{\xi}$, $i = 1, \ldots, n$, in scenario s, $s = 1, \ldots, S$. Natural requirements are:

$$\begin{aligned}
&\frac{1}{S}\sum_{s=1}^{S} \xi_i^s \approx \mu_i, && \forall i \\
&\frac{1}{S}\sum_{s=1}^{S} (\xi_i^s - \mu_i)(\xi_j^s - \mu_j) \approx \sigma_{ij}, && \forall i, j \\
&\frac{1}{S}\sum_{s=1}^{S} \frac{(\xi_i^s - \mu_i)^3}{\sigma_i^3} \approx 0, && \forall i \\
&\frac{1}{S}\sum_{s=1}^{S} \frac{(\xi_i^s - \mu_i)^4}{\sigma_i^4} \approx 3, && \forall i.
\end{aligned}$$

Note that, in order to simplify the model, we assume uniform probabilities π_s for each scenario. Approximate moment matching is obtained by minimizing the following squared error, with respect to the set of realized values ξ_i^s:

[9] See, e.g., [4] for a discussion of numerical quadrature, Gaussian quadrature, and low-discrepancy sequences, including Sobol sequences.

$$
\begin{aligned}
& w_1 \sum_{i=1}^{n} \left[\frac{1}{S} \sum_{s=1}^{S} \xi_i^s - \mu_i \right]^2 \\
& + w_2 \sum_{i=1}^{n} \sum_{j=1}^{n} \left[\frac{1}{S} \sum_{s=1}^{S} \left(\xi_i^s - \mu_i \right) \left(\xi_j^s - \mu_j \right) - \sigma_{ij} \right]^2 \\
& + w_3 \sum_{i=1}^{n} \left[\frac{1}{S} \sum_{s=1}^{S} \left(\frac{\xi_i^s - \mu_i}{\sigma_i} \right)^3 \right]^2 + w_4 \sum_{i=1}^{n} \left[\frac{1}{S} \sum_{s=1}^{S} \left(\frac{\xi_i^s - \mu_i}{\sigma_i} \right)^4 - 3 \right]^2 .
\end{aligned}
$$

The objective function includes four weights w_k, $k = 1, \ldots, 4$, which may be used to fine tune performance. It should be mentioned that the resulting scenario optimization problem need not be convex. However, if we manage to find any solution with a low value of the "error" objective function, this is arguably a satisfactory solution, even though it is not necessarily the globally optimal one. The idea can be generalized to "property matching," since we can match other features that are not necessarily related to moments; see, e.g., [19].

Moment (or property) matching has been criticized, since it is possible to build counterexamples, i.e., pairs of distributions that share the first few moments, but are actually quite different (see, e.g., [18]). An alternative idea is to rely on metrics fully capturing the distance between probability distributions. Thus, given a scenario tree topology, we should find the assignment of values and probabilities minimizing some distance with respect to the "true" distribution. Alternatively, given a large scenario tree, we could try to reduce it to a more manageable size by optimal scenario reduction. See, e.g., [17] for more details. We should mention that a counterargument to proponents of the optimal approximation approach is that we are not really interested in the distance between the ideal distribution and the scenario tree. What really matters is the quality of the solution. For instance, if we are considering a plain mean–variance portfolio optimization problem, it can be argued that matching the first two moments is all we need. See, e.g., [21, Chapter 4] for a discussion of these issues. Additional useful references are [23, 24].

We also remark a couple of possibly useful guidelines in shaping the structure of the scenario tree:

1. If the model just aims at a robust first-stage decision, it may be advisable to use a rich branching factor at the first stage, and a limited branching factor at later stages (provided that the tree is arbitrage-free when dealing with financial applications).
2. The number of stages can also be limited by using non-uniform time steps. For instance, the first two time steps could correspond to one month, and the later ones to one year.
3. A clever way to limit the number of stages is to augment the objective function with a term measuring the quality of the terminal state. By doing so, we may

avoid myopic decisions and reduce the size of the tree by limiting the number of stages (see, [14] for an application to ATO production planning).

8.1 In- and out-of-sample stability

Whatever scenario generation strategy we employ, it is important to check the stability of the resulting solution. There are quite sophisticated studies on the stability of stochastic optimization models, relying on formal metric spaces concepts. Here, we only consider a down-to-earth approach following the discussion in [21]. See, e.g., [26] for a formal analysis.

Let us denote the exact stochastic optimization problem, in a succinct way, by

$$\min_{\mathbf{x}\in S} \mathbb{E}_{\mathbb{P}}[f(\mathbf{x}, \tilde{\boldsymbol{\xi}})],$$

which involves the expectation of the objective function $f(\cdot,\cdot)$ under the true measure $\mathbb{P}$, i.e., the "exact" distribution of the vector $\tilde{\boldsymbol{\xi}}$ of risk factors. The actual problem that we solve is based on an approximate scenario tree $\mathcal{T}$ and can be denoted by

$$\min_{\mathbf{x}\in S} \widehat{f}(\mathbf{x}; \mathcal{T}),$$

where the notation $\widehat{f}$ emphasizes that we are just estimating the true expected value of the objective, given a generated scenario tree $\mathcal{T}$ that replaces the true measure $\mathbb{P}$. Each scenario tree induces an optimal solution of the approximate problem. Thus, if we sample trees $\mathcal{T}_i$ and $\mathcal{T}_j$, we obtain solutions $\mathbf{x}_i^*$ and $\mathbf{x}_j^*$, respectively, as well as corresponding values of the objective function. If the tree generation mechanism is reliable, we should observe a certain stability in the output solution. We may consider stability in the solution itself, but it is easier, and perhaps more relevant, to check stability in the value of the objective function. When the objective function is relatively flat, we may find rather different solutions with comparable performance. After all, if we want to minimize cost or maximize profit, the value of the objective is what matters most.

Two concepts of stability should be considered. **In-sample stability** means that, if we sample two scenario trees, the values of the solutions that we obtain should not be too different:

$$\widehat{f}(\mathbf{x}_i^*; \mathcal{T}_i) \approx \widehat{f}(\mathbf{x}_j^*; \mathcal{T}_j). \tag{45}$$

This definition does not apply directly, if the scenario tree is generated deterministically. In that case, we might compare trees with slightly different branching structures, in order to check whether the tree structure that we are using is rich enough to ensure solution stability. This concept of stability is called *in-sample*, as we evaluate a solution using the *same* tree that we have used to find the solution itself. But since the tree is only a limited representation of uncertainty, we should wonder what happens when we apply the solution in the real world, where a different

scenario may unfold. This leads to **out-of-sample stability**, where we compare the objective value that we obtain from the optimization model to the actual expected performance of the solution.

$$\mathbb{E}_{\mathbb{P}}[f(\mathbf{x}_i^*, \tilde{\boldsymbol{\xi}})] \approx \mathbb{E}_{\mathbb{P}}[f(\mathbf{x}_j^*, \tilde{\boldsymbol{\xi}})]. \tag{46}$$

If the trees are reliable, we should not notice a significant difference in performance. This kind of check is not too hard in two-stage models, as we should just plug the first-stage solution into several second-stage problems, each one corresponding to an observation of the risk factors $\tilde{\boldsymbol{\xi}}$. Typically, solving a large number of second-stage problems is not too demanding, especially if they are just linear programs; as a rule, it takes more time to solve one large stochastic LP than a large number of deterministic LPs.

Unfortunately, this is not so easy in a multistage setting, where realistic performance evaluation should be carried out by a costly rolling horizon simulation. To be more precise, we should consider two alternatives:

- In the **sliding** or **rolling horizon** approach,[10] we solve a multistage problem with H stages, apply the first-stage solution, observe an out-of-sample realization of the risk factors for the first time period, and solve again a multistage problem with H stages starting from the second time period of the original model.
- In the **shrinking horizon** approach, we solve a multistage problem with H stages, apply the first-stage solution, observe an out-of-sample realization of the risk factors, but then we solve a multistage problem with just $H - 1$ stages. At each time step, we reduce the number of stages.

The second approach makes sense when we really have a well-defined horizon for performance evaluation, and it is computationally cheaper than the first one. Anyway, while a rolling horizon is technically feasible, it is quite expensive computationally. A possible (cheaper) alternative is to generate solutions $\mathbf{x}_i^*$ and $\mathbf{x}_j^*$ from trees $\mathcal{T}_i$ and $\mathcal{T}_j$, respectively, and then to check the values of the (approximate) objective functions by swapping the trees. In other words, we may check whether

$$\widehat{f}(\mathbf{x}_i^*; \mathcal{T}_j) \approx \widehat{f}(\mathbf{x}_j^*; \mathcal{T}_i).$$

The need for out-of-sample evaluation shows a possible trouble with the stochastic programming approach. Stochastic programming provides us with a first-stage, here-and-now solution, but there is no obvious way to adapt decisions at later stages when uncertainty unfolds. All of the decisions are generated explicitly and are associated with nodes in the scenario tree. On the contrary, alternative approaches, like dynamic programming and decision rules (see the references in Section 9), provide us with decisions in feedback form, which makes them very suitable for out-of-sample simulation and evaluation of policies.

[10] The term *receding* horizon is also used in the model predictive control literature.

9 For further reading

In this chapter we have just scratched the surface of modeling by stochastic programming, sometimes using material adapted from [1, 5, 6]. In this section we provide some additional references on specific topics that we have to omit for the sake of space. An excellent source of additional material is `https://www.stoprog.org/resources`

9.1 Solution methods

Since stochastic optimization models, even in the linear case, can be large-scale problems, it may be useful to consider specifically tailored solution strategies. Usually, such solution strategies rely on some form of decomposition, which in turn takes advantage of structural properties, like the convexity of the recourse function. An early treatment of the subject is provided by [20]. See also [3] for additional topics and [28] for a formal treatment. We should also recall that functional formulations point out the fact that decision variables are actually functions, i.e., objects within an infinite-dimensional space. Scenario generation is a form of discretization to approximate the problem within a finite-dimensional framework. An alternative strategy, aimed at avoiding the exponential explosion of the scenario tree, relies on parameterized decision rules. See, e.g., [22].

9.2 Approximate dynamic programming

As we have seen, decisions in a multistage problem are functions of the whole observed sample path of the risk factors. However, quite often we may simplify the dependence by taking advantage of the Markov property. This amounts to saying that we may summarize all the information we need to make a decision by introducing a set of suitable state variables. This leads to the stochastic dynamic programming approach, which is quite related with nested formulations. Unfortunately, a literal application of the method is not quite practical, as it is plagued by the curse of dimensionality (unless we have a finite and small state space). Nevertheless, there is a wide range of approximate dynamic programming strategies that can be exploited to define near optimal decision strategies. See [7] for a short tutorial and [25] for a thorough treatment.

9.3 Stochastic vs. robust optimization

Since the topic of this chapter is stochastic optimization, we have assumed that we have reliable information about the probability distribution that we should use in scenario generation. If we do not want to rely on any probabilistic information, possibly because of lack of data, we may apply a different approach and associate with $\boldsymbol{\xi}$ an uncertainty set $\mathcal{U}$ (a compact subset of $\mathbb{R}^l$), and consider the worst-case robust model

$$\min_{\mathbf{x}\in S}\left[\max_{\boldsymbol{\xi}\in\mathcal{U}} f(\mathbf{x},\boldsymbol{\xi})\right]. \tag{47}$$

Quite often, the formulation of a static, worst-case robust optimization models leads to a tractable convex optimization problem.

The picture is less pleasing when we consider an adaptive model. For instance, the following model is one way to formulate a two-stage, adaptive worst-case optimization model in the linear case:

$$\begin{aligned}\min_{\mathbf{x},\mathbf{y}(\cdot)} \quad & \mathbf{c}^\mathsf{T}\mathbf{x} + \max_{\boldsymbol{\xi}\in\mathcal{U}} \mathbf{q}^\mathsf{T}\mathbf{y}(\boldsymbol{\xi}) \\ \text{s.t.} \quad & \mathbf{A}(\boldsymbol{\xi})\mathbf{x} + \mathbf{B}(\boldsymbol{\xi})\mathbf{y}(\boldsymbol{\xi}) \le \mathbf{h}(\boldsymbol{\xi}), \qquad \boldsymbol{\xi}\in\mathcal{U} \\ & \mathbf{x}\in\mathcal{X}, \mathbf{y}(\boldsymbol{\xi}) \ge \mathbf{0}.\end{aligned}$$

Just as in the stochastic case, second-stage variables $\mathbf{y}(\cdot)$ are actually functions, and we must define a suitable solution strategy based on some form of discretization,

A further approach, which is a kind of hybrid between the expected and the worst-case extremes, is to assume that the probability measure $\mathbb{P}$ is not known with certainty, but we can define an ambiguity set $\mathcal{P}$ of probability measures. This leads to the distributionally robust optimization (DRO) problem

$$\min_{\mathbf{x}\in S}\left\{\sup_{\mathbb{P}\in\mathcal{P}} \mathbb{E}_{\mathbb{P}}\left[f(\mathbf{x},\boldsymbol{\xi})\right]\right\}. \tag{48}$$

The theory behind DRO is rather demanding, as we need a suitable way to define the ambiguity set, hopefully leading to a tractable problem. See [2] and [29] for more information about robust optimization.

9.4 Surrogate optimization

When applying a stochastic optimization model, we assume the availability of reliable distributional information. If we do not assume that, we may resort to robust optimization models. However, we may be dealing with an even worse setting: the case in which we do not even have a formal model! When dealing with complex systems, the function $f(\mathbf{x},\boldsymbol{\xi})$, which maps decisions and realizations of risk factors into the performance of interest may not be given by an explicit expression, and

all we can do is to run a complex simulation model to get noisy estimates of the expected performance. This is a common setting in simulation-based optimization, which is often called black-box optimization.

Based on a set of sample paths (i.e., replications of the random simulation experiment), we are conceptually able to build a function of the design or control variables:

$$\widehat{F}(\mathbf{x}) \approx \mathbb{E}_{\mathbb{P}}\left[f(\mathbf{x}, \tilde{\boldsymbol{\xi}})\right].$$

In general, we cannot say much about the function $\widehat{F}(\mathbf{x})$, and its minimization is often accomplished by stochastic search methods like genetic algorithms or particle swarm optimization, which are typically not quite efficient, and certainly not a good choice when each simulation run is computationally efficient.

A possibly better strategy is to build a global representation of $\widehat{F}(\mathbf{x})$, i.e., a *metamodel* or *surrogate function*. Unlike the *local* approximations, like linear or quadratic models estimated by linear regression, that are typical of classical response surface methods, surrogate optimization relies on a *global* representation. Typical tools of the trade are radial basis functions and Gaussian process regression. The cheap surrogate function is used to wisely choose the values of $\mathbf{x}$ that we want to feed into the black box model in the search of the optimal solution. Excellent references on surrogate optimization are [11, 12, 15].

References

1. A. Alfieri, P. Brandimarte. Stochastic Programming Models for Manufacturing Applications. In: A. Matta, Q. Semeraro (eds.). *Design of Advanced Manufacturing Systems*. Springer, 2005.
2. D. Bertsimas, D. den Hertog. *Robust and Adaptive Optimization*. Dynamic Ideas, 2022.
3. J.R. Birge, F. Louveaux. *Introduction to Stochastic Programming (*2nd ed.*)*. Springer, 2011.
4. P. Brandimarte. *Numerical Methods in Finance and Economics: A MATLAB-Based Introduction* (2nd ed.). Wiley, 2006.
5. P. Brandimarte. *Quantitative Methods: An Introduction for Business Management*. Wiley, 2011.
6. P. Brandimarte. *An Introduction to Financial Markets: A Quantitative Approach*. Wiley, 2018.
7. P. Brandimarte. *From Shortest Paths to Reinforcement Learning. A MATLAB-Based Introduction to Dynamic Programming*. Springer, 2021.
8. G. Cornuejols, J. Peña, R. Tütüncü. *Optimization Methods in Finance* (2nd ed.). Cambridge University Press, 2018.
9. J. Curwin, R. Slater. *Quantitative Methods for Business Decisions* (6th ed.). Cengage Learning EMEA, 2008.
10. M. Fisher, A. Raman. Reducing the cost of demand uncertainty through accurate response to early sales. *Operations Research*, **44** (1996), 87–99.
11. A.I.J. Forrester, A. Sóbester, A.J. Keane. *Engineering Design via Surrogate Modeling: A Practical Guide*. Wiley, 2008.
12. R. Garnett. *Bayesian Optimization*. Cambridge University Press, 2023. See also `https://bayesoptbook.com`
13. A. Geyer, M. Hanke, A. Weissensteiner. No-arbitrage conditions, scenario trees, and multi-asset financial optimization. *European Journal of Operational Research*, **206** (2010), 609–613.
14. D.G. Gioia, E. Fadda, P. Brandimarte. Rolling horizon policies for multi-stage stochastic assemble-to-order problems. *International Journal of Production Research*, **62** (2023), 5108–5126.

15. R.B. Gramacy. *Surrogates: Gaussian Process Modeling, Design, and Optimization for the Applied Sciences*. CRC Press, 2020.
16. J.H. Hammond, A. Raman. *Sport Obermeyer Ltd*. Harvard Business School Case 695022-PDF-ENG, 1994.
17. H. Heitsch, W. Römisch. Scenario reduction algorithms in stochastic programming. *Computational Optimization and Applications*, **24** (2003), 187–206.
18. R. Hochreiter, G.C. Pflug. Financial scenario generation for stochastic multi-stage decision processes as facility location problems. *Annals of Operations Research*, **152** (2007), 257–272.
19. K. Hoyland, S.W. Wallace, Generating scenario trees for multistage decision problems. *Management Science*, **47** (2001), 296–307.
20. P. Kall, S.W. Wallace. *Stochastic programming*. Wiley, 1994. See also `https://www.stoprog.org/sites/default/files/files/manujw.pdf`
21. A.J. King, S.W. Wallace. *Modeling with Stochastic Programming*. Springer, 2012
22. D. Kuhn, W. Wiesemann, A. Georghiu. Primal and dual linear decision rules in stochastic and robust optimization. *Mathematical Programming*, **130** (2011), 177–209.
23. N. Löhndorf". An empirical analysis of scenario generation methods for stochastic optimization. *European Journal of Operational Research*, **255** (2016), 121–132.
24. S. Mehrotra, D. Papp. Generating moment matching scenarios using optimization techniques. *SIAM Journal of Optimization*, **23** (2013), 963–999.
25. W.B. Powell. *Approximate Dynamic Programming: Solving the Curses of Dimensionality (*2nd ed.*)*. Wiley, 2011.
26. W. Römisch. Stability of stochastic programming problems. In: A. Ruszczynski, A. Shapiro(eds.). *Stochastic programming*. Vol. 10 of Handbooks in Operations Research and Management Science. Elsevier, 2003.
27. A. Ruszczyński, A. Shapiro. Stochastic programming models. In: A. Ruszczynski, A. Shapiro (eds.). *Stochastic programming*. Vol. 10 of Handbooks in Operations Research and Management Science. Elsevier, 2003.
28. A. Shapiro, D. Dentcheva, A. Ruszczyński. Lectures on Stochastic Programming: Modeling and Theory. SIAM, 2009.
29. X.A. Sun, A.J. Conejo. *Robust Optimization in Electric Energy Systems*. Springer, 2021.
30. S.A. Zenios. *Practical Financial Optimization: Decision Making for Financial Engineers*. Blackwell Publishing, 2007.