AIMMS Modeling Guide - Inventory Control Problem

This file contains only one chapter of the book. For a free download of the complete book in pdf format, please visit www.aimms.com.

AIMMS B.V.
Diakenhuisweg 29-35
2033 AP Haarlem
The Netherlands
Tel.: +31 23 5511512

AIMMS Inc.
11711 SE 8th Street
Suite 303
Bellevue, WA 98005
USA
Tel.: +1 425 458 4024

AIMMS Pte. Ltd.
55 Market Street #10-00
Singapore 048941
Tel.: +65 6521 2827

AIMMS
SOHO Fuxing Plaza No.388
Building D-71, Level 3
Madang Road, Huangpu District
Shanghai 200025
China
Tel.: ++86 21 5309 8733

Email: info@aimms.com
WWW: www.aimms.com

Chapter 17

An Inventory Control Problem

This chapter

In this chapter you will encounter a multi-period inventory control problem with uncertain demand. At the beginning of each period the volume of production is decided prior to knowing the exact level of demand. During each period the demand becomes known, and as a result, the actual inventory can be determined. The objective in this problem is to minimize overall expected cost. The problem can be translated into a stochastic multi-stage optimization model. Such a model is a nontrivial extension of the two-stage model discussed in Chapter 16, and will be examined in detail. An alternative statement of the objective function is developed, and an instance of the stochastic inventory model is provided for illustrative purposes.

References

As already stated in Chapter 16, there is a vast literature on stochastic programming, but most of it is only accessible to mathematically skilled readers. Two selected book references on stochastic programming are [In94] and [Ka94].

Keywords

Linear Program, Stochastic Program, Multi-Stage, Control-State Variables, Mathematical Derivation, Worked Example.

17.1 Introduction to multi-stage concepts

This section

In this section the extension of two-stage stochastic programming to multi-stage programming is discussed. Tree-based terminology is used to characterize the underlying model structure.

Event parameters

Stochastic models contain so-called event parameters that do not have a priori fixed values. An event is an act of nature that falls outside the control of a decision maker. A typical example of an event parameter is product demand inside a decision model that determines production levels. The exact demand values are not known when the production decision has to be made, but they become known afterwards. Prior to the production decision, it only makes sense to consider various data realizations of the event parameter demand, and somehow take these into account when making the production decision.

Control and state variables

A stochastic model contains two types of variables, namely control and state variables. *Control variables* refer to all those decision variables that must be decided at the beginning of a period prior to the outcome of the uncertain event parameters. *State variables*, on the other hand, refer to all those decision variables that are to be determined at the end of a period after the outcome of the uncertain event parameters are known.

Two-stage versus multi-stage

The term *two-stage* decision making is reserved for the sequence of control decisions (first stage), event realizations, and state determination (second stage) for a single time period. The term *multi-stage* decision making refers to sequential two-stage decision making, and is illustrated in Figure 17.1.

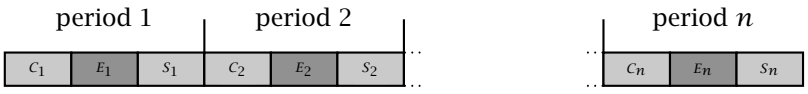

Figure 17.1: Multi-stage decision making

Assume few realizations

The number of possible data realizations of event parameters is usually very large. In theory, it is customary to relate the event parameters to continuous random variables with infinitely many outcomes. However, in practical applications it turns out better from a computational point of view to assume not only a finite number of data realizations, but also relatively few of them. This requires careful data modeling, so that the approximation with relatively few data realizations is still useful for decision making.

Tree with event probabilities

Whenever the underlying model is a multi-period model, the data realizations of event parameters can be expressed most conveniently in the form of a *tree*. An example is provided in Figure 17.2. Each level in the tree corresponds to a time slice, and each arc represents a particular realization of the event parameters. A node in the tree refers to a state of the system. The label associated with each arc is the *event description*. The fraction associated with each arc is the corresponding *event probability*. Note that the sum over all probabilities emanating from a single state equals one, reflecting the fact that all event parameter realizations are incorporated in the tree.

Conditional probabilities . . .

The event probabilities mentioned in the previous paragraph can be viewed as conditional probabilities. They describe the probability that a particular event realization will take place *given* the state from which this event emanates. In practical applications, these conditional probabilities form the most natural input to describe the occurrence of events.

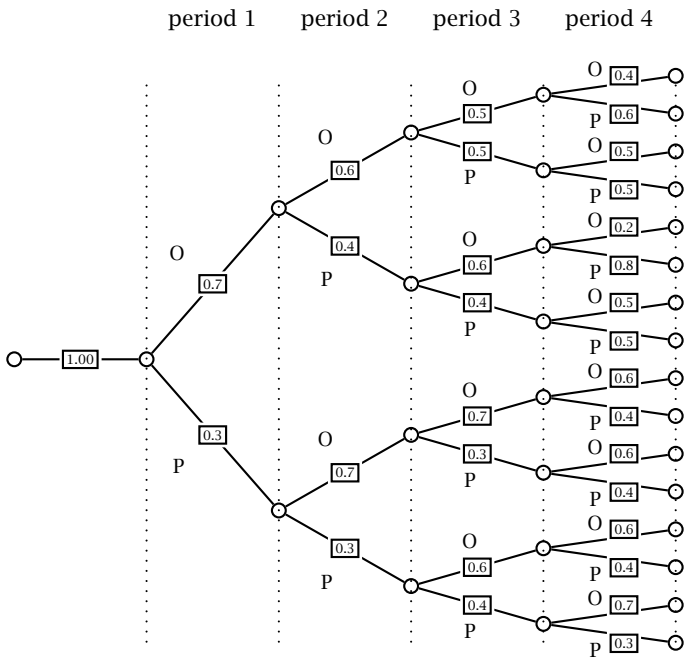

Figure 17.2: Tree with event labels and conditional event probabilities

... versus unconditional probabilities

By multiplying the event probabilities along the path from the root (the initial state) until a particular state, the unconditional probability to reach that particular state is computed. The tree with unconditional state probabilities is illustrated in Figure 17.3. Note that for all events in a single period, the sum over all unconditional probabilities add up to one. These probabilities will be used in the sequel to weight the outcome at each state to compute the overall expected outcome.

Scenarios

A *scenario* in the multi-stage programming framework is the collection of all event data instances along a path from the root to a particular leaf node in the event tree. Thus, the number of leaf nodes determines the number of scenarios. Note that the concepts of scenario and event coincide when there is only a single period. By definition, the probabilities associated with scenarios are the same as the unconditional probabilities associated with the leaf nodes (i.e. terminal states).

Two related terminologies

You may have noticed that there are two related terminologies that mingle naturally in the description of multi-stage stochastic programming models. One characterizes the concepts typically used in the stochastic programming literature, while the other one characterizes these same concepts in terms of tree structures. The tree terminology enables you to visualize concepts from stochastic programming in a convenient manner, and will be used throughout

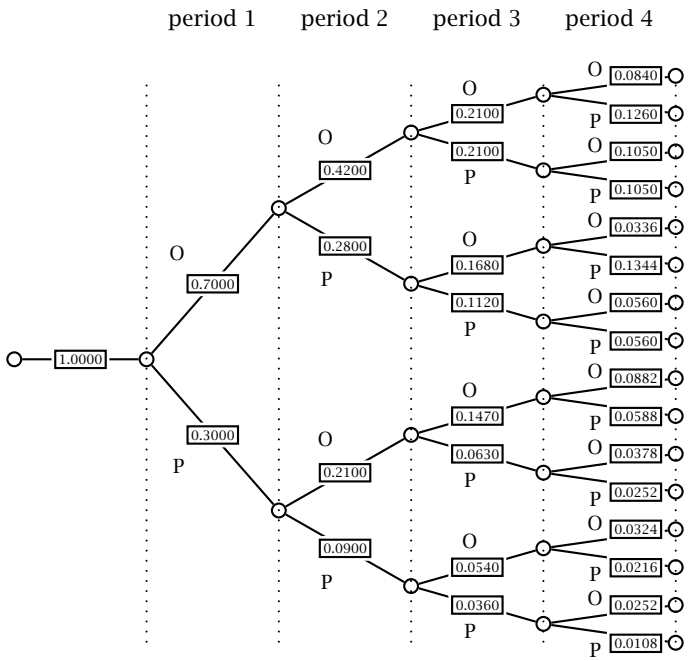

Figure 17.3: Tree with unconditional state probabilities

the chapter. The two related terminologies are summarized in Table 17.1.

stochastic terminology	tree-based terminology
event	arc
state	node
initial state	root node
final state	leaf node
scenario	path (root-to-leaf)

Table 17.1: Two related terminologies

17.2 An inventory control problem

This section

In this section you will encounter a simplified example in which the volume of beer to be bottled is decided while minimizing bottling, inventory and external supply costs. This example is used to illustrate the multi-stage approach to stochastic modeling discussed in the previous section.

Bottling of beer

Consider the decision of how much beer to bottle during a particular week. There are different beer types, and the available bottling machines can be used for all types. There is an overall bottling capacity limitation. Beer bottled in a particular week is available to satisfy demand of subsequent weeks. Bottling cost and storage cost are proportional to the amount of beer that is either bottled or stored. There is a minimum amount of storage required at the end of the last period.

Decide now and learn later

Demand is assumed to be uncertain due to major fluctuations in the weather. The decision variable of how much beer to bottle in a particular week is taken prior to knowing the demand to be satisfied. Therefore, the decision variable is a control variable, and demand is the event parameter. Once weekly demand becomes known, the inventory can be determined. Therefore, the inventory variable is a state variable. The term 'decide now and learn later' is frequently used in the literature, and reflects the role of the control variables with respect to the event parameters.

Demand scenarios

The uncertain demand over time can be characterized in terms of scenarios. For the sake of simplicity, assume that there are only a few of them. There will be pessimistic and optimistic events emanating from each state, and their conditional probabilities are known. All input data and model results are provided in Section 17.5.

17.3 A multi-stage programming model

This section

In this section a multi-stage programming model formulation of the inventory control problem of the previous section is developed in a step-by-step fashion, and the entire model summary is presented at the end.

Notation based on states

The model identifiers need to be linked to the scenario tree in some uniform fashion. By definition, all variables are linked to nodes: state variables are defined for every node, while control variables are defined for emanating nodes. It is therefore natural to index variables with a state index. Event parameters, however, are linked to arcs and not to nodes. In order to obtain a uniform notation for all identifiers, event parameters will also be indexed with a state index in exactly the same way as state variables. This is a natural choice, as all arcs correspond to a reachable state.

Bottling capacity constraint

The decision to bottle beer of various types during a particular time period is restricted by the overall available bottling capacity c during that period expressed in [hl]. Let b denote beer types to be bottled, and s the states to be considered. In addition, let x_s^b denote the amount of beer of type b to be bottled at emanating state s in [hl]. Note that this decision is only defined for emanating states and thus not for terminating states. To enforce this re-

striction, consider the element parameter α_s, which refers to the previous (or emanating) state of state s. When s refers to the initial state, the value of α_s is empty. For nonempty elements α_s the bottling capacity constraint can be written as follows.

$$\sum_b x^b_{\alpha_s} \leq c \qquad \forall \alpha_s$$

It is straightforward to implement element parameters, such as α_s, in the AIMMS language.

Inventory determination constraint

The inventory of each beer type at a particular reachable state, with the exception of the already known inventory at the initial state, is equal to the inventory at the emanating state plus the amount to be bottled decided at the emanating state plus externally supplied beer pertaining to the reachable state minus the demand pertaining to that reachable state. Note that externally supplied beer is used immediately to satisfy current demand, and will not exceed the demand due to cost minimization. Let y^b_s denote the amount of beer of type b that is stored at state s in [hl], and let z^b_s denote the external supply of beer of type b at state s in [hl]. Then, using d^b_s to denote the demand of beer of type b in state s in [hl], the inventory determination constraint can be written as follows.

$$y^b_s = y^b_{\alpha_s} + x^b_{\alpha_s} + z^b_s - d^b_s \qquad \forall (s,b) \mid \alpha_s$$

Inventory capacity constraint

Assume that the space taken up by the different types of bottled beer is proportional to the amount of [hl] bottled, and that total inventory space is limited. Let $\bar{y}$ denote the maximum inventory of bottled beer expressed in [hl]. Then the inventory capacity constraint can be written as follows.

$$\sum_b y^b_s \leq \bar{y} \qquad \forall s$$

Demand requirement constraint

In the previous inventory determination constraint there is nothing to prevent currently bottled beer to be used to satisfy current demand. However, as indicated in the problem description, the amount bottled in a particular period is only available for use during subsequent periods. That is why an extra constraint is needed to make sure that at each reachable state, inventory of each type of beer at the corresponding emanating state, plus the external supply of beer pertaining to the reachable state, is greater than or equal to the corresponding demand of beer. This constraint can be written as follows.

$$y^b_{\alpha_s} + z^b_s \geq d^b_s \qquad \forall (s,b)$$

Profit determination constraint

During each period decisions are made to contribute to overall profit. For each state, the contribution to profit is determined by considering sales revenues attached to this state minus the sum of all costs associated with this state. These costs cover the corresponding bottling cost, the external supply cost, and the inventory cost. Let ps^b denote the selling price of beer of type b in [$/hl], and let cp^b, ci^b, and ce^b denote the variable cost coefficients in [$/hl] associated with bottling, inventory and external supply. The entire state-specific contribution to profit, denoted by v_s, can now be written as follows.

$$v_s = \sum_b ps^b d_s^b - \sum_b (cp^b x_{\alpha_s}^b + ci^b y_s^b + ce^b z_s^b) \qquad \forall s$$

Objective function

In the presence of uncertain demand it does not make sense to talk about a deterministic overall profit determination. The approach in this section is to sum all state-specific profit contributions weighted with their unconditional probability of occurrence. It turns out that this computation expresses the expected profit level for the entire planning horizon, the length of which is equal to the number of time periods covered by the event tree. Such profit level can then be written as follows.

$$\sum_s p_s v_s$$

Alternative formulation

In the next section it is shown that the above expression coincides with the expected profit over all scenarios, where scenario-specific contributions to profit are weighted with their unconditional probability of occurrence. The number of terms in this alternative formulation of the objective function is then equal to the number of terminal states (i.e. the number of leaf nodes in the event tree). The two main ingredients are the scenario probabilities and the scenario profits. The scenario probabilities are the unconditional probabilities associated with the leaf nodes, and add up to one. The profit contribution per scenario is obtained by summing all corresponding state-specific profit contributions.

Verbal model summary

The above objective and the constraints that make up the stochastic programming formulation of the simplified inventory control problem can now be summarized through the following qualitative model formulation.

Maximize: *total expected net profit,*

Subject to:

- *for all emanating states: total bottling volume must be less than or equal to overall bottling capacity,*
- *for all beer types and reachable states: inventory at reachable state is equal to inventory at emanating state plus bottling volume at emanating state plus external supply pertaining to reachable state minus demand pertaining to reachable state,*

- *for all reachable states: inventory of bottled beer at reachable state must be less than or equal to maximum inventory of bottled beer,*
- *for all beer types and reachable states: inventory at emanating state plus external supply pertaining to reachable state must be greater than or equal to demand pertaining to reachable state, and*
- *for all reachable states: total net profit is sales revenue minus total costs consisting of bottling, inventory and external supply costs.*

Notation

The following symbols have been introduced to describe the objective function and the constraints.

Indices:

b	*beer types*
s	*states*

Parameters:

ps^b	*selling price of beer type b [$/hl]*
cp^b	*production cost of bottling beer type b [$/hl]*
ci^b	*inventory cost of storing beer type b [$/hl]*
ce^b	*external supply cost of beer type b [$/hl]*
c	*overall capacity to bottle beer [hl]*
$\bar{y}$	*maximum inventory of bottled beer [hl]*
d_s^b	*demand of beer type b in state s [hl]*
p_s	*probability of reaching state s [-]*
α_s	*previous state of state s in event tree*

Variables:

x_s^b	*beer type b bottled at emanating state s [hl]*
y_s^b	*beer type b stored at state s [hl]*
z_s^b	*external supply of beer b at state s [hl]*
v_s	*state-specific profit contribution at state s [$]*

Mathematical model summary

The mathematical description of the model can now be summarized as follows.

Maximize:

$$\sum_s p_s v_s$$

Subject to:

$$\begin{aligned}
\sum_b x^b_{\alpha_s} &\le c && \forall\, \alpha_s \\
y^b_{\alpha_s} + x^b_{\alpha_s} + z^b_s - d^b_s &= y^b_s && \forall\, (s,b) \mid \alpha_s \\
\sum_b y^b_s &\le \bar{y} && \forall\, s \\
y^b_{\alpha_s} + z^b_s &\ge d^b_s && \forall\, (s,b) \\
\sum_b ps^b d^b_s - \sum_b (cp^b x^b_{\alpha_s} + ci^b y^b_s + ce^b z^b_s) &= v_s && \forall\, s \\
x^b_s &\ge 0 && \forall\, (b,s) \\
y^b_s &\ge 0 && \forall\, (b,s) \\
z^b_s &\ge 0 && \forall\, (b,s)
\end{aligned}$$

17.4 Equivalent alternative objective function

This section

In this section you will find a proof of the fact that the expected profit function used in the previous section can be expressed in an alternative but equivalent form.

Recursive profits and probabilities

Consider the following identifiers defined over the set of states.

π_s	*conditional probability of event prior to state s*
p_s	*unconditional probability of reaching state s*
v_s	*state-specific profit contribution at state s [$]*
w_s	*cumulative profit contribution at state s [$]*

Then, the recursive relationships between these identifiers are defined for each node in the tree (starting at the root), and will be used in the sequel.

$$\begin{aligned}
p_s &= \pi_s p_{\alpha_s} \\
w_s &= v_s + w_{\alpha_s}
\end{aligned}$$

Initialization

The recursive unconditional probabilities emanate from the initial state probability, which is equal to one. The recursive cumulative profit levels emanate from the initial state profit level, which is assumed to be zero. These recursive relationships can be implemented inside AIMMS using a FOR statement inside a procedure.

Alternative formulation

Let $l = \{0, 1, \ldots\}$ denote a level in the event tree, and let $L(l)$ denote the set of all nodes corresponding to this level. In addition, let $\hat{l}$ denote the last level of the tree. Then the objective function expressing expected profit can be written in two seemingly different but equivalent forms.

$$\sum_s p_s v_s = \sum_{s \in L(\hat{l})} p_s w_s$$

This equality turns out to be a special instance of the following theorem.

Theorem

Let the symbol $\bar{l}$ denote any particular level of the event tree. Then the following equality holds for each $\bar{l}$.

$$\sum_{l=0}^{\bar{l}} \sum_{s \in L(l)} p_s v_s = \sum_{s \in L(\bar{l})} p_s w_s$$

Corollary

Note that when $\bar{l}$ is equal to $\hat{l}$, then the term on the left of the equal sign is nothing else but $\sum_s p_s v_s$, and the alternative formulation follows directly from the theorem.

Proof

The proof will be carried out by induction on the number of levels $\bar{l}$. For $\bar{l} = 0$, the theorem holds trivially. Consider any particular $\bar{l} > 0$, and assume that the theorem holds for $\bar{l} - 1$. Then to prove that the theorem also holds for $\bar{l}$, you need to rewrite summations, use the above recursive definitions of p_s and w_s, use the fact that $\sum_{k \mid \alpha_k = s} \pi_k = 1$, and, of course, use the statement of the theorem for $\bar{l} - 1$ during the last step. All this is done in the following statements.

$$\begin{aligned}
\sum_{s \in L(\bar{l})} p_s w_s &= \sum_{s \in L(\bar{l})} p_s (v_s + w_{\alpha_s}) \\
&= \sum_{s \in L(\bar{l})} p_s v_s + \sum_{s \in L(\bar{l})} p_s w_{\alpha_s} \\
&= \sum_{s \in L(\bar{l})} p_s v_s + \sum_{s \in L(\bar{l}-1)} \sum_{k \mid \alpha_k = s} \pi_k p_{\alpha_k} w_{\alpha_k} \\
&= \sum_{s \in L(\bar{l})} p_s v_s + \sum_{s \in L(\bar{l}-1)} p_s w_s \sum_{k \mid \alpha_k = s} \pi_k \\
&= \sum_{s \in L(\bar{l})} p_s v_s + \sum_{s \in L(\bar{l}-1)} p_s w_s \\
&= \sum_{s \in L(\bar{l})} p_s v_s + \sum_{l=0}^{\bar{l}-1} \sum_{s \in L(l)} p_s v_s \\
&= \sum_{l=0}^{\bar{l}} \sum_{s \in L(l)} p_s v_s
\end{aligned}$$

□

17.5 A worked example

This section

In this section an input data set is provided together with an overview of the results based on the multi-stage programming model developed in Section 17.3. The initial probabilities are the same as in Figures 17.2 and 17.3. The revenues

and cost values are presented in Table 17.2 while the demand requirements for both beer types are listed in Figure 17.4.

b	ps^b [\$/hl]	cp^b [\$/hl]	ci^b [\$/hl]	ce^b [\$/hl]
light	300	12.0	5.0	195.0
regular	400	10.0	5.0	200.0

Table 17.2: Revenues and cost values

The overall bottling capacity c is 46.0 [hl] and the maximum inventory of bottled beer $\bar{y}$ is 52.0 [hl]. Initial stock for light beer is 17 [hl], and initial stock for regular beer is 35 [hl].

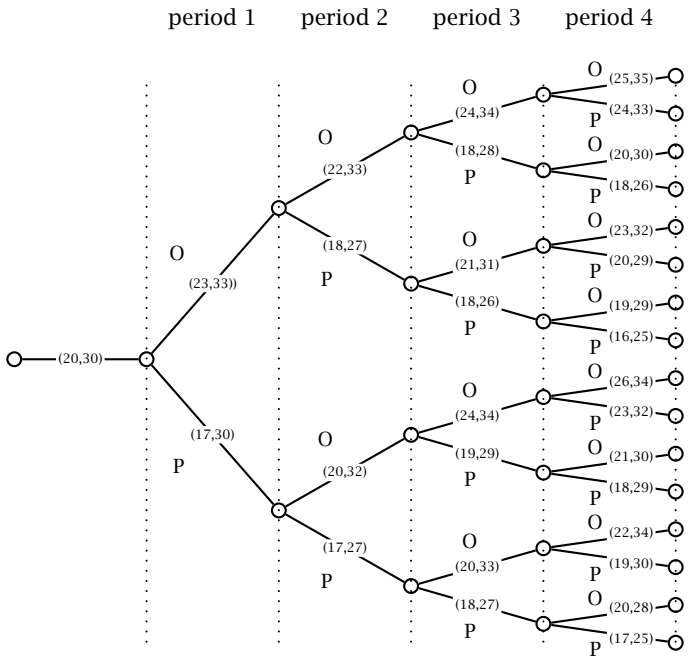

Figure 17.4: Demand requirements for both beer types (light,regular)

Optimal solution

The optimal solution of the multi-stage programming model is presented in Table 17.3 with a total weighted profit of 76482.4 [\$]. Notice that the hierarchical structure of the scenarios is not only reflected in the order and description of their labels, but also in the zero-nonzero pattern of the control and state variables. Even though optimal decisions are determined for each period and all possible scenarios, in most practical applications only the initial decisions are implemented. The model is usually solved again in each subsequent period after new scenario information has become available.

Binding constraints

Once you have implemented the model yourself, you may want to verify that the production capacity constraint is binding and has a positive shadow price for scenarios 'I', 'O', 'OO', 'OP' and 'PO'. Similarly, the storage capacity constraint is binding for scenarios 'PP' and 'PPP'. This indicates that both types of capacity play a vital role in the optimal use of the brewery.

b	light				regular				
s	x_s^b	y_s^b	z_s^b	d_s^b	x_s^b	y_s^b	z_s^b	d_s^b	v_s
I	18.0	17.0	20.0	20.0	28.0	35.0	30.0	30.0	7840.0
O	18.0	18.0	6.0	23.0	28.0	30.0		33.0	18194.0
OO	18.0	18.0	4.0	22.0	28.0	28.0	3.0	33.0	17694.0
OOO		18.0	6.0	24.0		28.0	6.0	34.0	17704.0
OOOO			7.0	25.0			7.0	35.0	18735.0
OOOP			6.0	24.0			5.0	33.0	18230.0
OOP		18.0		18.0		28.0		28.0	15874.0
OOPO			2.0	20.0			2.0	30.0	17210.0
OOPP				18.0		2.0		26.0	15790.0
OP	19.0	18.0		18.0	27.0	31.0		27.0	15459.0
OPO		19.0	3.0	21.0		27.0		31.0	17387.0
OPOO			4.0	23.0			5.0	32.0	17920.0
OPOP			1.0	20.0			2.0	29.0	17005.0
OPP		19.0		18.0		32.0		26.0	15047.0
OPPO				19.0		3.0		29.0	17285.0
OPPP		3.0		16.0		7.0		25.0	14750.0
P	18.0	18.0		17.0	27.0	33.0		30.0	16349.0
PO	17.0	18.0	2.0	20.0	29.0	28.0		32.0	17694.0
POO		17.0	6.0	24.0		29.0	6.0	34.0	17706.0
POOO			9.0	26.0			5.0	34.0	18645.0
POOP			6.0	23.0			3.0	32.0	17930.0
POP		17.0	1.0	19.0		29.0	1.0	29.0	16181.0
POPO			4.0	21.0			1.0	30.0	17320.0
POPP			1.0	18.0				29.0	16805.0
PP	19.0	19.0		17.0	26.0	33.0		27.0	15154.0
PPO		19.0	1.0	20.0		26.0		33.0	18292.0
PPOO			3.0	22.0			8.0	34.0	18015.0
PPOP				19.0			4.0	30.0	16900.0
PPP		20.0		18.0		32.0		27.0	15452.0
PPPO				20.0		4.0		28.0	17180.0
PPPP		3.0		17.0		7.0		25.0	15050.0

Table 17.3: Optimal solution

17.6 Summary

In this chapter a multi-stage stochastic programming model was viewed as a sequence of two-stage stochastic programming models. A tree-based terminology was used to describe event probabilities and multi-stage scenarios. All concepts were illustrated on the basis of a simplified inventory control model. Two alternative and seemingly different objective functions were introduced, and were shown to be equivalent. A complete input data set was provided,

together with an overview of the model results.

Exercises

17.1 Implement the multi-stage mathematical program summarized at the end of Section 17.3 using the example data provided in Section 17.5. Verify that the optimal solution found with AIMMS coincides with the one presented in Table 17.3.

17.2 Implement the model with the alternative objective function described in Section 17.4, and verify whether the optimal solution remains the same.

17.3 Set up an experiment in AIMMS to investigate the sensitivity of the overall optimal stochastic programming objective function value to changes in the number of scenario's.

Bibliography

[In94] G. Infanger, *Planning under uncertainty*, Boyd & Fraser, Danvers, Massachusetts, 1994.

[Ka94] P. Kall and S.W. Wallace, *Stochastic programming*, John Wiley & Sons, Chichester, 1994.