Stochastic Programming (SP) with EMP

1 Introduction

This chapter[1] describes the stochastic programming (SP) extension of GAMS EMP. We build a stochastic model based on a deterministic model by defining model parameters to be uncertain. Then GAMS EMP replaces these uncertain parameters by random variables. The distribution of the random variables is controlled by the user. Note that these random variables are not variables in the sense of mathematical optimization, but they can be understood as random parameters.

The chapter is organized as follows: In section 2 the basic principles are introduced with the well known news vendor problem, in section 3 a multi-stage model is discussed, in section 4 a class of models with chance constraints is presented, while how to model risk measures is the topic of section 5. In section 6 a summary of keywords and possible solver configurations is given and finally more details on scenarios and output extraction follow in section 7.

2 The News Vendor

Suppose a news vendor wants to maximize his profit z. He has to decide how many newspapers to buy from a distributor to satisfy demand d. He buys x newspapers at a cost of c per unit. He manages to sell s newspapers at a price of v per paper. If the demand is less than his expectations $(d < x)$, then the number i of left-over newspapers are stored in an inventory at a holding cost of h per unit. If the demand exceeds his expectations $(d > x)$, then there will be l customers whose demand cannot be met and a penalty of p per unit of unsatisfied demand has to be paid. Assuming that the demand is known the optimization problem can be expressed mathematically as follows:

$$\begin{array}{ll} \text{Max}_{x,s,i,l} & z = vs - cx - hi - pl \\ \text{s.t.} & d = s + l \\ & i = x - s \\ & x, s, l, i \geq 0. \end{array} \tag{2.1}$$

The objective is to maximize the profit z. Here x, s, i, l are the decision variables and the demand d is a known parameter. The demand d equals the sum of the satisfied demand s (papers sold) and the unsatisfied demand l (lost sales). The inventory i equals the difference between the number of papers bought x and those sold s. Note that $x, s, l, i \geq 0$.

This can easily be translated into GAMS. We assume that the values for c, p, h, v and d are given. Note that our mathematical formulation is case-sensitive but the GAMS code is not .

```
Scalar
        c       Purchase costs per unit             / 30 /
        p       Penalty shortage cost per unit      / 5 /
        h       Holding costper unit leftover       / 10 /
        v       Revenue per unit sold               / 60 /
*       Random parameters
        d       Demand                              / 63 /;

Variable Z Profit;
Positive Variables
```

[1] This version created by Martha Loewe and Michael Ferris in June 2013.

```
         X Units bought
         I Inventory
         L Lost sales
         S Units sold;

Equations Profit, Row1, Row2;

* Profit, to be maximized
Profit..  Z =e= v*S - c*X - h*I - p*L;
* demand = UnitsSold + LostSales
Row1.. d =e= S + L;
* Inventory = UnitsBought - UnitsSold
Row2.. I =e= X - S;

Model nb / all /;
solve nb max Z use lp;
```

Since the demand d is known there is no uncertainty in this model and the optimal solution is obvious: the best decision is to buy exactly as many newspapers as are demanded. Now we move to more realistic assumptions and consider several examples where the demand is not known from the outset.

2.1 Uncertain demand: discrete distribution

Consider the case when the buying decison should be made *before* a realization of the demand d becomes known. In our example, the news vendor has to buy newspapers in the morning without knowing the precise demand. However, given his experience he expects the demand to be 45 in 70% of all cases, 40 with a probability of 20% and 50 with a probability of 10%. One way to model such a situation is to regard the demand D as a *random variable*. By D, we denote the demand when viewed as a random variable in order to distinguish it from a particular realization d. We assume that the probability distribution of D can be estimated from experience or historical data and is therefore known. Suppose that the set of all realizations of D is finite and given by Ω:

$$\Omega = \{d_1, d_2, ..., d_{|\Omega|}\}.$$

Each realization of D is called a *scenario*. So there is a finite set of scenarios, each associated with probability p_j, where $\sum_j p_j = 1$. In our example, the random variable D takes the values $d_1 = 45$, $d_2 = 40$ and $d_3 = 50$ with respective probabilities $p_1 = 0.7$, $p_2 = 0.2$ and $p_3 = 0.1$. The decision x has to be made before the realization of the demand D is known.

After the news vendor has made the decision of how many newspapers x to buy on a particular day the actual demand is disclosed. He may have bought more that he can sell and may have to store the surplus in his inventory or he may not have bought enough and some demand may remain unsatisfied. We can translate this situation into a model with two stages: in stage 1 a decision x is made without knowing the future, then one realization of the future unfolds and in stage 2 a second period decision s, i, l is made that attempts to react to the new situation. The action taken in stage 2 is called *recourse*. The key idea in this approach is the evolution of information.

We express this two stage model mathematically in the following way :

$$\text{Max}_x \quad Z(x,D) = -cx + \mathbb{E}[Q(x,D)], \quad x \geq 0, \tag{2.2}$$

where

$$\begin{aligned} Q(x,D) = \text{Max}_{s,i,l} \quad & vs_D - hi_D - pl_D \\ \text{s.t.} \quad & x - s_D - i_D \qquad = 0 \\ & s_D \qquad + l_D = D \\ & s_D, i_D, l_D \geq 0. \end{aligned} \tag{2.3}$$

Given the uncertainty of the demand we aim to maximize the *expected value* of the profit, denoted by $\mathbb{E}[Z(x,D)]$. The expected value of the profit is the profit *on average*. Note that since we have a finite number of scenarios and their probabilities

1st stage decision variable	2nd stage decision variable $\Omega = \{d_1, d_2, d_3\}$	Probabilities
x	$y_{d_1} = (s_{d_1}, l_{d_1}, i_{d_1})$ $y_{d_2} = (s_{d_2}, l_{d_2}, i_{d_2})$ $y_{d_3} = (s_{d_3}, l_{d_3}, i_{d_3})$	Scenario 1: $p_1 = 0.7$ Scenario 2: $p_2 = 0.2$ Scenario 3: $p_3 = 0.1$

Table 31.1: Two stages in news vendor problem

are known, the expected value of the profit $\mathbb{E}[Z(x,D)]$ can be expressed as a weighted sum:

$$\begin{aligned}\mathbb{E}[Z(x,D)] &= -cx + \mathbb{E}[Q(x,D)] \\ &\quad -cx + \sum_{k=1}^{3} p_k Q(x,d_k).\end{aligned} \tag{2.4}$$

Notice that we have separated the original optimization problem into two different problems that are solved at two stages. The decision variable in the first problem (at stage 1) is x and it is the vector $y_D = (s_D, i_D, l_D)$ in the second problem (at stage 2). We introduce additional notation to explicitly represent the underlying structure of the problem. Let

$$q_D = \begin{bmatrix} v \\ -h \\ -p \end{bmatrix}, \quad T_D = \begin{bmatrix} I \\ 0 \end{bmatrix}, \quad W_D = \begin{bmatrix} -I & -I & 0 \\ I & 0 & I \end{bmatrix}, \quad h_D = \begin{bmatrix} 0 \\ D \end{bmatrix}.$$

Using this new notation the second stage optimization problem from (2.3) becomes:

$$\begin{aligned} Q(x,D) = \text{Max}_{y_D} \quad & q_D^T y_D \\ \text{s.t.} \quad & T_D x + W_D y_D = h_D, \quad y_D \geq 0. \end{aligned} \tag{2.5}$$

Note that in our specific example $T_D \equiv T$, $W_D \equiv W$ and $q_D \equiv q$ since they are independent of the demand D. The *general* two stage optimization problem is given here:

$$\begin{aligned} \text{Max}_{x \in \mathbb{R}^n} \quad & Z(x,\zeta) = -c^T x + \mathbb{E}[Q(x,\zeta)] \\ \text{s.t.} \quad & Ax = b, \quad x \geq 0, \end{aligned} \tag{2.6}$$

where

$$\begin{aligned} Q(x,\zeta) = \text{Max}_{y \in \mathbb{R}^m} \; & q_\zeta^T y \\ \text{s.t.} \quad & T_\zeta x + W_\zeta y = h_\zeta, \quad y \geq 0, \end{aligned} \tag{2.7}$$

and

$$\zeta = (q, h, T, W),$$

the data of the second stage problem. Note that in our news vendor example there are no first stage equations $Ax = b$. Table 31.1 summarizes the two stages in our example. The problem as stated in (2.6), (2.7) can be converted to a single large scale optimization problem. The extended form is given below:

$$\begin{aligned} \text{Max}_x \quad & -c^T x + \textstyle\sum_i p_i [\text{Max}_y q_{\zeta_i}^T y] \\ \text{s.t.} \quad & Ax = b \\ & T_\zeta x + W_\zeta y = h_\zeta \\ & x, y \geq 0. \end{aligned} \tag{2.8}$$

Finally, we present the problem in matrix form, a notation that has the advantage of explicitly displaying the structure of the second stage:

$$\begin{aligned} &\text{Max}_{x,y} \quad -c^T x + \textstyle\sum_i p_{\zeta_i} q_{\zeta_i}^T y \\ &\text{s.t.} \quad \begin{pmatrix} A & & & & \\ T_{\zeta_1} & W_{\zeta_1} & & & \\ T_{\zeta_2} & & W_{\zeta_2} & & \\ \vdots & & & \ddots & \\ T_{\zeta_m} & & & & W_{\zeta_m} \end{pmatrix} \begin{pmatrix} x \\ y_{\zeta_1} \\ y_{\zeta_2} \\ \vdots \\ y_{\zeta_m} \end{pmatrix} = \begin{pmatrix} b \\ h_{\zeta_1} \\ h_{\zeta_2} \\ \vdots \\ h_{\zeta_m} \end{pmatrix}, \quad x, y_{\zeta_i} \geq 0. \end{aligned} \tag{2.9}$$

To formulate this 2-stage-model in GAMS we introduce the notions of a *core problem* and *annotations*. The *core problem* defines q, h, T, W as parameters instead of having them as random variables. It can be written as follows:

$$\begin{array}{ll} \text{Max}_{x,y} & -c^T x + q^T y \\ \text{s.t.} & \begin{pmatrix} A & 0 \\ T & W \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} b \\ h \end{pmatrix}, \quad x, y \geq 0. \end{array} \tag{2.10}$$

The *annotations* give details about the various realizations of the random variable ζ, the scenarios.

For our news vendor example the core model is precisely the model given on page 595. Then we add the annotations: the distribution of the random variable (in our case the demand which is part of the variable h) has to be specified as random and every variable and equation must be assigned to one of the two stages. This information is introduced by writing a text file named %emp.info%. Here is the file for our model:

```
file emp / '%emp.info%' /; put emp '* problem %gams.i%'/;
$onput
randvar d discrete 0.7 45 0.2 40 0.1 50
stage 2 I L S d
stage 2 Row1 Row2
$offput
putclose emp;
```

First, we define the parameter `d` to be a random variable (`randvar`) with a `discrete` distribution: with probability 0.7 it takes a value of 45, with probability 0.2 it takes a value of 40, and with probability 0.1 it takes a value of 50. Then the variables and equations of `stage 2` are listed. The variables and equations not listed in the %emp.info% file are automatically assigned to a stage, with a default assumed to be stage 1. Note that the objective variable (in our case Z) and the profit equation are thus assigned to the highest stage specified (stage 2 in this example). Observe that Z is in fact a random variable since it is a function of the random variable D. As such it cannot be maximized, EMP implicitly maximizes the *expected value* of Z. We believe this might lead to some confusion since the expected value of Z belongs to `stage 1`. We show later how to specify more clearly the fact that we are maximizing $\mathbb{E}(Z)$. All keywords that can be used in the emp.info file are introduced in subsequent examples and summarized in section 6 on page 620.

Given the probability distribution the solvers of stochastic programming models create various scenarios and evaluate them. Where should the details of these scenarios be stored (or communicated to the modeler)? In deterministic models the modeler does not need to specify how the output should be stored. Using EMP for stochastic programming the modeler can store the results for each scenario in standard parameters. Here is how it is done:

```
Set scen        Scenarios / s1*s3 /;
Parameter
    s_d(scen)   Demand realization by scenario
    s_x(scen)   Units bought by scenario
    s_s(scen)   Units sold by scenario;

Set dict / scen .scenario.''
           d    .randvar .s_d
           s    .level   .s_s
           x    .level   .s_x /;

solve nb max z use emp scenario dict;
```

The size of the set `scen` defines the maximal number of scenarios we are willing to store results for. The three dimensional set `dict` contains mapping information between symbols in the model (in the first position) and symbols to store solution information (in the third position), and the type of storing (in the second position). An exception to this rule is the tuple with label `scenario` in the second position. This tuple determines the symbol (in the first position) that is used as the scenario index. This scenario symbol can be a multidimensional set. A tuple in this set represents a single scenario. In our example, we want to store the realization for each sceanrio for the random variable `d` in the parameter `s_d` and the levels of the variables `s` and `x` in the parameters `s_s` and `s_x` respectively. A more detailed description on scenarios and how this set works can be found in section 7.

Finally, the `solve` statement needs to be adjusted: we use `emp` instead of `lp` and add `scenario dict` to indicate that a stochastic problem should be solved.

2.2 Uncertain demand: continuous distribution

Now let's assume that the random variable D has a continuous distribution, say a Normal distribution with mean 45 and standard deviation 10. This problem has the same structure as the problem discussed in section 2.1, where the random variable D has a discrete distribution. The difference is that the set Ω, that is the set of all realizations of D, contains an infinite number of scenarios. There are various ways of modeling this, one of which is a sampling procedure implemented in the solver. A finite number of scenarios is generated to approximate Ω and thus the problem is converted to a problem as discussed in section 2.1.

In GAMS we can model a continuous distribution of the random variable by changing the `randvar` line in the emp.info file in the following way:

```
randvar d normal 45 10
```

Note that currently only the solver LINDO has implemented a sampling procedure for parametric distributions. More details about sampling are given in the next section. In Table 31.2 all parametric distributions that can be modeled are listed.

Distribution	Parameter 1	Parameter 2	Parameter 3
Beta	shape 1	shape 2	
Cauchy	location	scale	
Chi_Square	deg. of freedom		
Exponential	lambda		
F	deg. of freedom 1	deg. of freedom 2	
Gamma	shape	scale	
Gumbel	location	scale	
Laplace	mean	scale	
Logistic	location	scale	
LogNormal	mean	std dev	
Normal	mean	std dev	
Pareto	scale	shape	
StudentT	deg. of freedom		
Triangular	low	mid	high
Uniform	low	high	
Weibull	shape	scale	
Binomial	n	p	
Geometric	p		
Hyper_Geometric	total	good	trials
Logarithmic	p-factor		
Negative_Binomial	failures	p	
Poisson	lambda		

Table 31.2: Parametric distributions supported by LINDO

We have to inform GAMS that we want the solver LINDO to be used. There are two ways to do this: we either state in the command line `emp=lindo` or we insert the following statement before the `solve` statement in the GAMS file :

```
option emp = lindo;
```

2.3 Sampling

Currently only the solver LINDO has the ability to perform sampling for continuous distributions. It generates 6 samples by default. There are three ways to customize sampling: 1) adding additional information to the emp.info file, 2) generating

a sample with the LINDO library `lsadclib` and then using this sample in any solver with EMP capabilities and 3) setting various options in the solver LINDO. We will discuss each method in more detail in the remainder of this section.

2.3.0.1 Customizing sampling with EMP Observe that customizing sampling with EMP is currently experimental, feedback is requested, and possible changes might occur when its notation becomes fixed. Currently, EMP provides two keywords to enable users to customize sampling: `sample` and `setSeed`. The keyword `sample` allows the user to customize the size of the sample in the emp.info file. Consider the following example:

```
randvar d normal 45 10
sample d 9
```

The second line determines the size of the sample of the distribution of the random variable D to be 9. Note that currently the LINDO Sampling library is used for this sampling. Users who don't have a valid LINDO license are limited to the Normal and Binomial distributions with a maximum sample size of 10.

The keyword `sample` also offers the possibility to determine a mathematical variance reduction method to be applied. Variance reduction is a procedure used to increase the precision of the estimated values from the distribution. LINDO provides three methods for reducing the variance: Monte Carlo sampling, Latin Square sampling and Antithetic sampling.

Consider a stochastic model with four random variables: E, F, G and H. Assume that E follows a Normal distribution with mean 23 and standard deviation 5, F follows a Normal distribution with mean 37 and standard deviation 8, G is uniformly distributed on the interval $[0,1]$ and H is binomially distributed with $n = 100$ and $p = 0.55$. We want for each random variable a sample size of 10 and we want to apply the three variance reduction methods in the following way: LINDO shall use Antithetic sampling for E and F, Monte Carlo sampling for G and Latin Square sampling for H. To model this we insert the following lines in the emp.info file:

```
1 randvar e normal 23 5
2 randvar f normal 37 8
3 randvar g uniform 0 1
4 randvar h binomial 100 55
5 sample e f 10 method1
6 sample g 12 method2
7 sample h 8 method3
```

In lines 1 to 4 the random variables and their distributions are defined, in lines 5 to 7 details about the sampling procedures are given. Note that the keyword `sample` can take more than one random variable (line 5) if the sample size and the variance reduction method for these random variables are the same. We need to add the following lines before the `solve` statement to specify the content of `method1`, `method2` and `method3` (we assume that the name of the model is `nb`):

```
1 $onecho > lindo.opt
2 SVR_LS_ANTITHETIC=method1
3 SVR_LS_MONTECARLO=method2
4 SVR_LS_LATINSQUARE=method3
5 $offecho
6 nb.optfile=1;
```

If the Latin Square sampling should also be used for E and F, we would simply change the emp.info file to replace the label `method 1` with the label `method 3`. For more details on variance reduction methods please consult the LINDO manual.

A further way to customize the sampling procedure is the keyword `setSeed`:

```
setSeed <seed>
```

This sets the seed for the random number generator of the sampling routines called using the `sample` keyword. If `setSeed` is used in the emp.info file, the seed is set once before we generate all samples. Please note that `setSeed` only works with a valid LINDO license.

2.3.0.2 Separating sampling and solving A user may want to sample from a distribution with the LINDO system and solve the model with another solver, say DE. This is possible with the sampling routines from the LINDO library `lsadclib`. We could solve the news vendor model by first drawing a sample from a Normal distribution with mean 45 and standard deviation 10 and then using the sample in the emp.info file. Here is the GAMS code for sampling from a Normal distribution where the sample size is 9:

```
$funclibin msllib lsadclib

function         setSeed                    / msllib.setSeed         /
                 sampleNormal    / msllib.sampleLSNormal  /
                 getSampleValues / msllib.getSampleValues /;

scalar k;
k = sampleNormal(45,10,9);

set g /1*9/; parameter sv1(g);
loop(g,
   sv1(g) = getSampleValues(k);
);
display sv1;
```

The directive in the first line makes the LINDO sampling library available, `msllib` is the internal library name. For further details and a list of the available distributions please consult the LINDO manual.

In the following lines we demonstrate how the sample is used in the emp.info file:

```
file emp / '%emp.info%' /; put emp '* problem %gams.i%'/;
put 'randvar d discrete '; loop(g, put (1/card(g)) ' ' sv1(g) ' ');
$onput
stage 2 I L S d
stage 2 Row1 Row2
$offput
putclose emp;
```

Line 2 states that the random variable D follows a discrete distribution and the probabilites and values are taken from the previously generated sample. The other lines of the emp.info file remained unchanged.

2.3.0.3 Sampling options in the LINDO solver There are some customizable sampling options in LINDO. The user could control the number of sampled scenarios by setting any of the following LINDO/SP options in the lindo.opt file:

```
STOC_NSAMPLE_PER_STAGE      - list of sample sizes per stage (starting at stage 2)
STOC_NSAMPLE_SPAR           - common sample size per stochastic parameter
STOC_NSAMPLE_STAGE          - common sample size per stage
```

For example, we could insert the following three lines before the `solve` statement:

```
option emp = lindo;
$echo STOC_NSAMPLE_STAGE = 100 > lindo.opt
nb.optfile = 1;
```

The first line tells GAMS to solve the `emp` modeltype using the LINDO solver, the second line writes "STOC_NSAMPLE_STAGE = 100" to the lindo.opt file, which indicates the solver to generate 100 samples per stage, and the third line informs GAMS to use the solver option file (i.e. lindo.opt). For more details about LINDO options please consult the user manual.

3 Multistage Models

In models with more than two stages the same principle applies as in 2-stage models: new information is revealed at the beginning of the stage and recourse decisions or adjustments are made after this information is available. At the point where

decisions are made only outcomes of the current stage and previous stages are available. In (3.11) this logic is pictured schematically.

$$\underbrace{\text{Make decision}}_{\text{Stage 1}} \rightarrow \underbrace{\text{Random Variable is realized} \rightarrow \text{Make decision}}_{\text{Stage 2}} \rightarrow \cdots \rightarrow \underbrace{\text{Random Variable is realized} \rightarrow \text{Make decision}}_{\text{Stage n}} \tag{3.11}$$

Observe that random variables which are realized in stage k are fixed parameters in stage $k+1$; stage 1 random variables are in fact simply given deterministic values.

Consider an inventory problem where the decision must be made at the end of each week how many hats should be bought in order to satisfy the stochastic demand in the following week with the aim to maximize the profit. We assume that the stochastic demand can be modeled using a Gamma distribution. The planning horizon is 3 weeks. *Before* the first week starts an initial purchase decision has to be made and the goods are stored in the inventory for use in week 1. At this point only the *distribution* of the demand of the first week is known. During the first week the *actual* demand is revealed and some items that were stored in the inventory are sold. Some items may be left over; they are stored as the inventory for the second week. In addition, a purchase decision for the second week has to be made given the size of the inventory and the *distribution* of the demand in the second week. Again, the *actual* demand is revealed in the course of the second week. The same holds for the third week.

We will model the problem with 4 stages, where the first stage corresponds to the preparation time before the first week, the second stage corresponds to decisions made in the first week, the third stage corresponds to decisions made the second week and the fourth stage corresponds to decisions made in the third week. Let t denote the stages. Note that while the stages range from $t=1$ to $t=4$, demand variables are realized only at $t=2$ to $t=4$. Let y_t be the amount bought and i_t the amount stored at the end of each stage and let D_t denote the demand in each week and s_t the amount sold each week. Note that we denote the demand with a capital D since it is a random variable. Let $\alpha=10$ be the cost per hat bought, $\beta=20$ the revenue per hat sold and $\delta=4$ the storage cost per unit. Further, in the storage facility a maximum of $\kappa=5000$ hats can be stored.

As discussed in section 2.2 above the solvers use a sampling procedure to approximate a problem with a continuous random variable such as a Gamma distributed random variable by a problem with a discrete distribution. Figure 31.1 illustrates the stages assuming a sample size of 6.

The stochastic optimization problem can be expressed as follows:

$$\begin{array}{lll} \text{Max}_{s_t,y_t,i_t} & z = -\alpha y_1 - \gamma i_1 + \mathbb{E}[\text{Max} \quad \beta s_2(D_2) - \alpha y_2(D_2) - \delta i_2(D_2) + \cdots + \\ & \mathbb{E}[\text{Max} \quad \beta s_4(D_4) + -\alpha y_4(D_4) - \delta i_4(D_4)] \ldots] \\ \text{s.t.} & i_1 = y_1 \\ & i_{t-1} + y_t = s_t + i_t \\ & s_t \leq i_{t-1} \\ & s_t \leq D_t \\ & i_t \leq \kappa \\ & y_1, i_1, s_t, y_t, i_t \geq 0, \end{array} \tag{3.12}$$

where $t = 2,\ldots,4$ and D_t follows a Gamma distribution. Note that s_t, y_t and i_t depend on the realization of D_t.

The GAMS code follows.

```
set t     "stages" /1*4/;
set st(t) "stages where sales occur" /2*4/ ;

scalars
    kappa    capacity of storage building
    alpha  cost per unit bought
    beta    revenue per unit sold
    delta   cost per unit stored at the end of time period;

parameters
    k    "shape of demand (1st parameter of gamma distribution)"
```

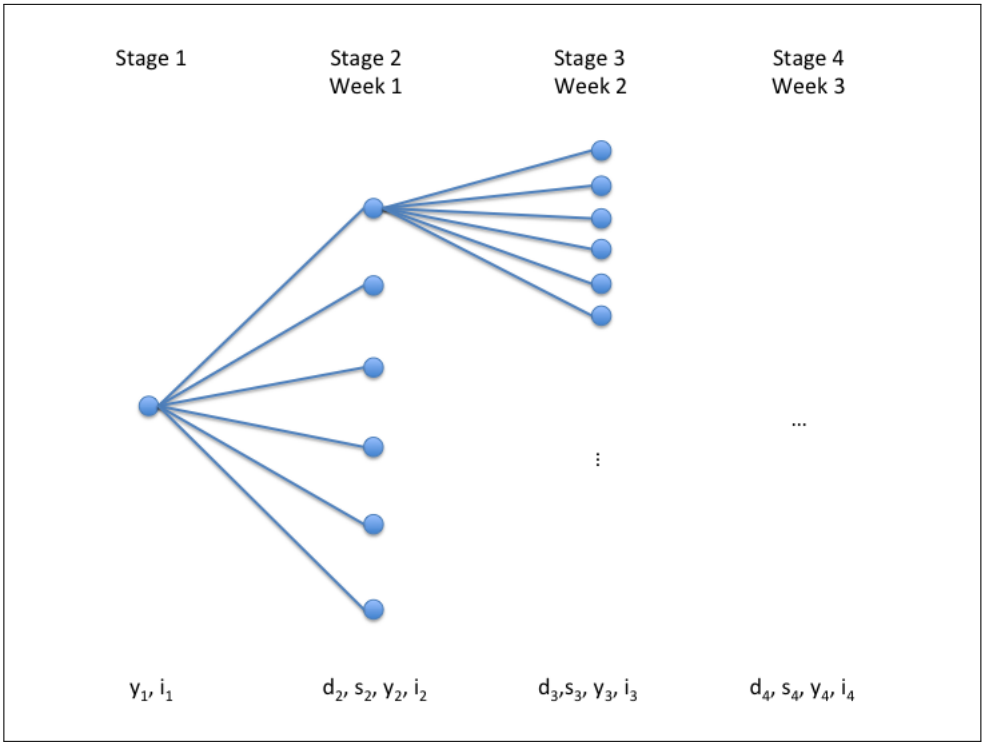

Figure 31.1: Stages in the inventory model

```
    d_theta(st)    "scale of demand (2nd parameter of gamma distribution)"
    d(st)    "demand";

positive variables
    y(t)  units to be bought in  time period t
    i(t)  ending inventory in period t
    s(t)  units sold in time period t;
free variable profit;

* Now assign all the data
k = 16;
parameter d_theta(st) / 2 208.3125, 3 312.5, 4 125 /;
d(st) = k * d_theta(st);
display d;
kappa = 5000;
alpha = 10;
beta = 20;
delta = 4;

equations
    profit_eq    profit to be max
    Bal_eq(t)    balance equation
    Sales_eq1(st)    sales cannot exceed demand
    Sales_eq2(t)    sales cannot exceed inventory of previous time period;

profit_eq..  profit =e= sum(t, beta * s(t)$st(t) - alpha * y(t) - delta * i(t));

Bal_eq(t)..  i(t-1) + y(t) =e= s(t)$st(t) + i(t) ;

Sales_eq1(st)..  s(st) =l= d(st);

```

```
Sales_eq2(t)$st(t)..  s(t) =l= i(t-1);

i.up(t) = kappa;
model inventory /all/;

file emp / '%emp.info%' /; put emp '* problem %gams.i%'/;
put emp; emp.nd=6;
put "randvar d('2') gamma ", k d_theta('2') /;
put "randvar d('3') gamma ", k d_theta('3') /;
put "randvar d('4') gamma ", k d_theta('4') /;
$onput
stage 1 y('1') i('1') Bal_eq('1')
stage 2 y('2') d('2') s('2') i('2') Bal_eq('2') Sales_eq1('2') Sales_eq2('2')
stage 3 y('3') d('3') s('3') i('3') Bal_eq('3') Sales_eq1('3') Sales_eq2('3')
stage 4 y('4') d('4') s('4') i('4') Bal_eq('4') Sales_eq1('4') Sales_eq2('4')
$offput
putclose emp;

Set scen        Scenarios / s1*s1000 /;
Parameter
    s_d(scen,st)   Demand realization by scenario
    s_y(scen,t)   Units bought by scenario
    s_s(scen,t)   Units sold by scenario
    s_i(scen,t)        Units stored by scenario;

Set dict /
  scen .scenario.''
  d  .randvar .s_d
  s  .level   .s_s
  y  .level   .s_y
  i  .level   .s_i  /;

solve inventory max profit using emp scenario dict;

display s_d, s_s, s_y, s_i;
```

Observe that in the core problem the values of the demand `d(t)` are replsced by the expected values of the random variable D_t which follows a Gamma distribution. Note that as expected, in stage 1 we have only the variables `y` and `i`, but no variables `s` and `d`. Note that currently only the solver LINDO can solve models with parametric distributions (see sections 2.2 and 2.3).

As discussed in section 2.3 above the user could use the LINDO system to generate samples and then solve the optimization problem with another solver. Here is the GAMS code that allows the user to draw samples for the inventory model (the sample size is 10). In this code we generate samples from 3 different distributions `f`, `g` and `h` and store the samples in the parameters `sv1`, `sv2` and `sv3`.

```
$funclibin msllib lsadclib

function setSeed         / msllib.setSeed         /
         sampleGamma      / msllib.sampleLSGamma   /
         getSampleValues / msllib.getSampleValues /;

$if not set ss $set ss 10
scalar f, g, h;
f = sampleGamma(16,208.3125,%ss%);
g = sampleGamma(16,312.5,%ss%);
h = sampleGamma(16,125,%ss%);

set j /1*%ss%/;
parameter sv1(j),sv2(j), sv3(j);

```

```
16 loop(j,
17    sv1(j) = getSampleValues(f); );
18 display sv1;
19
20 loop(j,
21    sv2(j) = getSampleValues(g); );
22 display sv2;
23
24 loop(j,
25    sv3(j) = getSampleValues(h); );
26 display sv3;
```

The user has to modify the `emp.info` file to use the generated samples (`sv1`, `sv2` and `sv3`) with probabilites $\frac{1}{|J|}$ to solve the model:

```
1  file emp / '%emp.info%' /; put emp '* problem %gams.i%'/;
2  put emp; emp.nd=6;
3  put "randvar d('2') discrete "; loop(j, put (1/card(j)) ' ' sv1(j) ' ');
4  put "randvar d('3') discrete "; loop(j, put (1/card(j)) ' ' sv2(j) ' ');
5  put "randvar d('4') discrete "; loop(j, put (1/card(j)) ' ' sv3(j)' ');
6  $onput
7  stage 1 y('1') i('1') Inv1_eq
8  stage 2 y('2') d('2') s('2') i('2') Bal_eq('2') Sales_eq1('2') Sales_eq2('2')
9  stage 3 y('3') d('3') s('3') i('3') Bal_eq('3') Sales_eq1('3') Sales_eq2('3')
10 stage 4 y('4') d('4') s('4') i('4') Bal_eq('4') Sales_eq1('4') Sales_eq2('4')
11 $offput
12 putclose emp;
```

Note that lines 3-5 are newly inserted in order to use the generated samples; the stages remain unchanged.

In case the modeler wants to use large samples with limited computing capability, LINDO offers the option to approximate the solution with a Benders decompostion algorithm. The following lines can be inserted to achieve this:

```
1 $onecho > lindo.opt
2 STOC_MAX_NUMSCENS = 1000000
3 STOC_NSAMPLE_STAGE = 40
4 STOC_METHOD = 1
5 $offecho
6 inventory.optcr=1e-4;
```

In line 2 we ensure that the maximum number of scenarios is large enough, the option in line 3 states the sample size and the option in line 4 determines the stochastic method to be used (1 means Nested Benders decomposition). For further details on LINDO options please consult the LINDO user manual. We add line 6 to ensure that the gap between the true solution and the approximation is reasonably small.

4 Chance Constraints

A major class of problems in stochastic modeling involves chance constraints. The goal in these problems is to make an optimal decision prior to the realization of random data while allowing the constraints to be violated with a certain probability.

Let $\tilde{A} \in \mathbb{R}^m \times \mathbb{R}^n$ be a random matrix and let $\tilde{b} \in \mathbb{R}^m$ be a random vector. Then a general stochastic linear program can be written as:

$$\begin{array}{ll} \text{Min}_x & c^T x \\ \text{s.t.} & \tilde{A}x \leq \tilde{b} \\ & x \geq 0. \end{array} \tag{4.13}$$

All feasible solutions satisfy all constraints simultaneously. The distinctive feature of a program with chance constraints is that we require $\tilde{A}x \leq \tilde{b}$ to be satisfied not in all cases but only with some prescribed probability p, where $0 < p \leq 1$. Note

that often we are interested in the *risk tolerance* ε, ε being the probability that a constraint is *not* satisfied. So $p = 1-\varepsilon$. Throughout this chapter we use p and $1-\varepsilon$ interchangeably. A general stochastic linear problem with chance constraints can be written as follows:

$$\begin{array}{ll} \text{Min}_x & c^T x \\ \text{s.t.} & P(\tilde{A}x \leq \tilde{b}) \geq p \\ & x \geq 0. \end{array} \tag{4.14}$$

Solvers convert a problem like (4.14) into a mixed-integer problem (MIP) first and then solve the MIP equivalent. They introduce a vector with binary variables, say $y_k \in \mathbb{R}^m$, for each scenario $k \in S$. The binary variables take value 1 if the corresponding constraint is satisfied in a scenario and 0 otherwise. A scenario-based formulation of the chance-constrained stochastic linear program (4.14) can be written as:

$$\begin{array}{ll} \text{Min}_x & c^T x \\ \text{s.t.} & A^k x \leq b^k + M^k(1-y_k) \\ & \sum_{k\in S} y_k \geq p \times |S| \\ & x \geq 0, \quad y \in (0,1)^{|S|}, \end{array} \tag{4.15}$$

where $M^k \in \mathbb{R}^m$ is a chosen big-M vector. The entries of the vector M^k should be chosen such that it does not cut off any feasible solution if an entry of $y_k = 0$.

We will first discuss the special case where the random matrix $\tilde{A}$ is a one-row vector $\tilde{a} \in \mathbb{R}^n$, the random vector $\tilde{b}$ is a single random variable $\tilde{b}$ and we have only one chance constraint. Then we will move on to the two ways to model problems with multiple chance constraints: using *joint chance constraints* and using *individual chance constraints*. Joint chance constraints require all constraints to be satisfied simultaneously with a given probability. Individual chance constraints require each constraint to be satisfied with a given probability independent of other constraints. We discuss joint chance constraints in section 4.2, individual chance constraints in section 4.3 and compare them in section 4.4. Finally, in section 4.5 the option to penalize violation of constraints is introduced.

4.1 Single Chance Constraints

Consider the following chance-constrained stochastic linear problem:

$$\begin{array}{ll} \text{Min}_x & c^T x \\ \text{s.t.} & P(\tilde{a}x \leq \tilde{b}) \geq p \\ & x \geq 0, \end{array} \tag{4.16}$$

where $\tilde{a}$ is a random row vector and $\tilde{b}$ is a random variable. Given a set of scenarios S let each scenario $k \in S$ be realized with probability π^k. The corresponding realizations of $\tilde{a}$ and $\tilde{b}$ are denoted by a^k and b^k respectively. The inequalities of the $|S|$ scenarios may be expressed as follows:

$$\begin{array}{l} a^1 x \leq b^1 \\ a^2 x \leq b^2 \\ \vdots \\ a^{|S|} x \leq b^{|S|} \end{array} \tag{4.17}$$

If each scenario is equally likely to be realized then the decision variable x must be chosen such that the inequality is satisfied in at least $p \times |S|$ scenarios.

Here is an example with two decision variables and 4 scenarios where each scenario is equally likely to be realized (i.e. $\pi^k = \frac{1}{4}$) :

$$\begin{array}{lll} \text{Min} & x_1 + x_2 & \\ \text{s.t.} & P(\omega x_1 + x_2 \geq 7) \geq 0.75, & \omega \in \Omega = \{1,2,3,4\} \\ & x_1, x_2 \geq 0. & \end{array} \tag{4.18}$$

Note that in this example b is fixed at 7. Note further that there are four scenarios since there are four possible realizations of ω. Since $p = 0.75$ and each scenario is equally likely to be realized we need to choose x_1 and x_2 such that the inequality

is satisfied in at least 3 scenarios. The inequalities for the four scenarios are given below:

$$\begin{array}{lll} k=1: & \omega^1=1 & \omega^1 x_1+x_2\geq 7 \\ k=2: & \omega^2=2 & \omega^2 x_1+x_2\geq 7 \\ k=3: & \omega^3=3 & \omega^3 x_1+x_2\geq 7 \\ k=4: & \omega^4=4 & \omega^4 x_1+x_2\geq 7. \end{array} \tag{4.19}$$

The MIP equivalent is given below:

$$\begin{array}{ll} \text{Min} & x_1+x_2 \\ \text{s.t.} & 1x_1+x_2\geq 7-M(1-y_1) \\ & 2x_1+x_2\geq 7-M(1-y_2) \\ & 3x_1+x_2\geq 7-M(1-y_3) \\ & 4x_1+x_2\geq 7-M(1-y_4) \\ & \\ & cc_1=1-\sum_k \pi^k y_k, \quad k=1,\ldots,4,\ \pi^k=\frac{1}{4} \\ & \\ & x_1,x_2\geq 0 \\ & 0\leq cc_1\leq (1-0.75) \\ & y_k\in(0,1). \end{array} \tag{4.20}$$

Observe that the first four constraints cover the four possible scenarios with ω taking the values 1, 2, 3 and 4 respectively. On the right handside we introduce a big-M factor and y_k, a binary indicator variable. y_k takes the value 1 if the constraint is satisfied and 0 otherwise. A new variable, cc_1, is introduced in the fifth constraint representing the probability that the constraint is violated. If $cc_1=0$ the sum equals 1, indicating that the constraint is satisfied in all four scenarios. If $cc_1=0.25$ the constraint remains unsatisfied in one scenario out of four (for this scenario $y_k=0$).

The problem can be modeled in GAMS as follows:

```
Scalar
        om  / 1 /;

Variable          Z     Objective;
Positive Variables X1, X2;

Equations OBJ, E1;

OBJ.. Z =e= X1 + X2;
E1..  om*X1 +   X2 =g=  7;

Model sc / all /;

file emp / '%emp.info%' /; put emp '* problem %gams.i%'/;
$onput
randvar om discrete 0.25 1  0.25 2   0.25 3   0.25 4
chance E1 0.75
$offput
putclose emp;

Set scen        scenarios / s1*s12 /;
Parameter
    s_om(scen)
    x1_l (scen)
    x2_l (scen)
    x1_m (scen)
    e1_l (scen);

Set dict / scen .scenario.''
           om  .randvar .s_om
```

```
31            x1   .level   .x1_l
32            x2   .level   .x2_l
33            x1   .marginal.x1_m
34            e1   .level   .e1_l/;
35
36  solve sc min z use emp scenario dict;
37  display s_om, x1_l, x2_l, x1_m, e1_l;
```

As introduced before we start with a core model and then add annotations and output handling information. Currently, there are no stages unlike in the problems with recourse. Observe that we introduced the new keyword `chance`. The line `chance E1 0.75` specifies that the constraint E1 must hold for at least 75% of all scenarios. We can verify that this requirement has been enforced by checking in the output file the level value of the constraint `e1_l`, and seeing the first scenario constraint is violated at the solution of the chance constrained problem.

Note that the default value of M in the solvers Lindo and DE is 10000. Currently it can only be customized in DE. We could insert the following five lines before the solve statement to set the value of M to 1000:

```
option emp = de;
$onecho > de.opt
ccreform bigM 1e3
$offecho
sc.optfile = 1;
```

Alternatively, the first line could be replaced by placing `emp=de` on the command line. Note that it is important that `optca` and `optcr` are assigned the appropriate values. The GAMS default for the absolute gap `optca` is 0 and the default for the relative gap `optcr` is 0.1.

Observe that in addition to converting the chance-constraint problem to a MIP using M the solver DE offers two further options to solve chance-constraint problems: a reformulation using a convex hull and a reformulation using indicator variables and indicator constraints. The following line indicates that a convex hull with $M = 1000$ and $\varepsilon = 0.00001$ is to be used:

```
ccreform cHull 1e3 1e-6
```

For indicator variables and constraints we use the following line:

```
ccreform indic
```

Note that currently only the solver CPLEX supports indicator variables, so the resulting reformulated problem has to be solved with CPLEX.

4.2 Joint Chance Constraints

In this section we discuss the general stochastic linear problem with chance constraints as introduced in (4.14) assuming that p is the predescribed probability that all constraints are simultaneously satisfied.

$$\begin{array}{ll} \text{Min}_x & c^T x \\ \text{s.t.} & P(\tilde{A}x \leq \tilde{b}) \geq p \\ & x \geq 0. \end{array} \tag{4.21}$$

Consider the example (4.18) from the previous section extended by one constraint, so that we have two decision variables and two constraints. The random data follow discrete uniform distributions so each scenario is equally likely.

$$\begin{array}{ll} \text{Min} & x_1 + x_2 \\ \text{s.t.} & P(\omega_1 x_1 + x_2 \geq 7;\ \omega_2 x_1 + 3x_2 \geq 12) \geq 0.6, \qquad (\omega_1, \omega_2) \in \Omega \\ & x_1, x_2 \geq 0, \end{array} \tag{4.22}$$

where

$$\Omega = \{(1,1),(1,2),(1,3),(2,1),(2,2),(2,3),(3,1),(3,2),(3,3),(4,1),(4,2),(4,3)\} \tag{4.23}$$

and

$$\pi^k = \pi(\omega_1, \omega_2) = \frac{1}{12} \quad \text{for all} \quad (\omega_1, \omega_2) \in \Omega. \tag{4.24}$$

Note that in this example b is not random and we have 12 scenarios that are all equally likely. The MIP equivalent is given below:

$$\begin{array}{ll} \text{Min} & x_1 + x_2 \\ \\ \text{s.t.} & 1x_1 + x_2 \geq 7 - M(1 - y_1) \\ & 1x_1 + 3x_2 \geq 12 - M(1 - y_1) \\ & 2x_1 + x_2 \geq 7 - M(1 - y_2) \\ & 1x_1 + 3x_2 \geq 12 - M(1 - y_2) \\ & \vdots \\ & 4x_1 + x_2 \geq 7 - M(1 - y_{12}) \\ & 3x_1 + 3x_2 \geq 12 - M(1 - y_{12}) \\ \\ & cc_1 = 1 - \sum_k \pi^k y_k, \quad k = 1, \ldots, 12, \pi^k = \frac{1}{12} \\ & x_1, x_2 \geq 0 \\ & 0 \leq cc_1 \leq (1 - 0.6) \\ & y_k \in (0, 1). \end{array} \tag{4.25}$$

Note that the first set of constraints cover the 12 scenarios, where each scenario has two constraints. The other constraints are similar to those introduced in (4.20).

The corresponding GAMS model follows.

```
Scalar
        om1  / 1 /
        om2  / 1 /;

Variable           Z     Objective;
Positive Variables X1,X2;

Equations OBJ, E1, E2;

OBJ.. Z =e= X1 + X2;
E1..  om1*X1 +   X2 =g=  7;
E2..  om2*X1 + 3*X2 =g= 12;

Model sc / all /;

file emp / '%emp.info%' /; put emp '* problem %gams.i%'/;
$onput
randvar om1 discrete 0.25 1  0.25 2  0.25 3  0.25 4
randvar om2 discrete 0.3333 1  0.3334 2  0.3333 3
chance E1 E2 0.6
$offput
putclose emp;

Set scen        scenarios / s1*s12 /;
Parameter
    s_om1(scen)
    s_om2(scen)
    x1_l (scen)
    x2_l (scen)
    x1_m (scen)
    e1_l (scen)
    e2_l (scen);

Set dict / scen .scenario.''
           om1  .randvar .s_om1
```

```
            om2   .randvar .s_om2
            x1    .level   .x1_l
            x2    .level   .x2_l
            x1    .marginal.x1_m
            e1    .level   .e1_l
            e2    .level   .e2_l/;

solve sc min z use emp scenario dict;
display s_om1, x1_l, x2_l, x1_m, e1_l, e2_l;
```

The line `chance E1 E2 0.6` specifies that in at least 60% of all scenarios both E1 and E2 must be satisfied at the same time. There are a total of 12 scenarios, so both constraints must be satisfied in at least 8 $(12 * 0.6 = 8)$ scenarios. We can verify that this requirement has been enforced by checking in the output file the level values of the constraints, i.e. `e1_l` and `e2_l`. Indeed, in the optimal solution both constraints hold in scenarios 4 to 12, so there are 9 scenarios that satisfy both inequalities.

4.3 Individual Chance Constraints

In this section we discuss the general stochastic linear problem with chance constraints assuming that there is no correlation between the probabilities of the rows of the matrix $\tilde{A}$:

$$\begin{array}{ll} \text{Min}_x & c^T x \\ \text{s.t.} & P(\tilde{A}_i.x \leq \tilde{b}_i) \geq p_i, \quad i = 1,\ldots,m \\ & x \geq 0. \end{array} \tag{4.26}$$

Consider the example from the previous section, this time with individual chance constraints and extended by one constraint:

$$\begin{array}{lll} \text{Min} & x_1 + x_2 & \\ \text{s.t.} & P(\omega_1 x_1 + x_2 \geq 7) \geq 0.75, & \omega_1 \in \Omega_1 = \{1,2,3,4\} \\ & P(\omega_2 x_1 + 3x_2 \geq 12) \geq 0.6, & \omega_2 \in \Omega_2 = \{1,2,3\} \\ & P(\omega_1 x_1 + \omega_2 x_2 \geq 10) \geq 0.5, & (\omega_1,\omega_2) \in \Omega_1 \times \Omega_2 = \Omega \\ & x_1, x_2 \geq 0. & \end{array} \tag{4.27}$$

Note that Ω is defined as in (4.23) above, we have again 12 scenarios each with probability $\pi^k = \frac{1}{12}$. However, in this example the first inequality must hold in 9 out of 12 scenarios $(0.75 * 12 = 9)$, the second inequality must hold in 8 out of 12 inequalities $(0.6 * 12 = 8)$ and the third inequality must hold in 6 out of 12 scenarios. Note further that we may have four types of scenarios: scenarios where all constraints are violated, scenarios where two constraints are violated, scenarios where one constraint is violated and scenarios where all three constraints are satisfied. The only condition is that for each constraint there is the respective number of scenarios where the constraint is satisfied. Note further that the random data in the third inequality is a combination of the random data of the first two inequalities. The inequalities for the scenarios are given below:

$$\begin{array}{lll} k = 1: & \omega_1^1 = 1,\ \omega_2^1 = 1 & \omega_1^1 x_1 + x_2 \geq 7 \\ & & \omega_2^1 x_1 + 3x_2 \geq 12 \\ & & \omega_1^1 x_1 + \omega_2^1 x_2 \geq 10 \\ k = 2: & \omega_1^2 = 1,\ \omega_2^2 = 2 & \omega_1^2 x_1 + x_2 \geq 7 \\ & & \omega_2^2 x_1 + 3x_2 \geq 12 \\ & & \omega_1^2 x_1 + \omega_2^2 x_2 \geq 10 \\ \vdots & & \\ k = 12: & \omega_1^{12} = 4,\ \omega_2^{12} = 3 & \omega_1^{12} x_1 + x_2 \geq 7 \\ & & \omega_2^{12} x_1 + 3x_2 \geq 12 \\ & & \omega_1^{12} x_1 + \omega_2^{12} x_2 \geq 10 \end{array} \tag{4.28}$$

The MIP equivalent follows:

$$
\begin{array}{ll}
\text{Min} & x_1 + x_2 \\
\\
\text{s.t.} & 1x_1 + x_2 \geq 7 - M(1-y_1^1) \\
& 1x_1 + 3x_2 \geq 12 - M(1-y_1^2) \\
& 1x_1 + 1x_2 \geq 10 - M(1-y_1^3) \\
& 2x_1 + x_2 \geq 7 - M(1-y_2^1) \\
& 1x_1 + 3x_2 \geq 12 - M(1-y_2^2) \\
& 2x_1 + 1x_2 \geq 10 - M(1-y_2^3) \\
& \vdots \\
& 4x_1 + x_2 \geq 7 - M(1-y_{12}^1) \\
& 3x_1 + 3x_2 \geq 12 - M(1-y_{12}^2) \\
& 4x_1 + 3x_2 \geq 10 - M(1-y_{12}^3) \\
\\
& cc_1 = 1 - \sum_k \pi^k y_k^1, \quad k = 1,\ldots,12, \quad \pi^k = \frac{1}{12} \\
& cc_2 = 1 - \sum_k \pi^k y_k^2, \quad k = 1,\ldots,12, \quad \pi^k = \frac{1}{12} \\
& cc_3 = 1 - \sum_k \pi^k y_k^3, \quad k = 1,\ldots,12, \quad \pi^k = \frac{1}{12} \\
\\
& x_1, x_2 \geq 0 \\
& 0 \leq cc_1 \leq (1 - 0.75) \\
& 0 \leq cc_2 \leq (1 - 0.6) \\
& 0 \leq cc_3 \leq (1 - 0.5) \\
& y_k^j \in (0,1).
\end{array}
\tag{4.29}
$$

As expected, there are three constraints for every scenario. Note that we introduce three new variables, cc_1, cc_2 and cc_3 and three corresponding constraints. Each of the variables has a different range mirroring the different probabilities with which a constraint may be violated.

The core model of the GAMS code is very similar to the core model with joint chance constraints. We just add the third inequality:

```
Equations OBJ, E1, E2, E3;

OBJ.. Z =e= X1 + X2;
E1..  om1*X1 +   X2 =g=  7;
E2..  om2*X1 + 3*X2 =g= 12;
E3..  om1*X1 + om2*X2 =g= 10;
```

There is a slight modification in the annotations:

```
file emp / '%emp.info%' /; put emp '* problem %gams.i%'/;
$onput
randvar om1 discrete 0.25 1   0.25 2  0.25 3  0.25 4
randvar om2 discrete 0.3333 1  0.3334 2  0.3333 3
chance E1 0.75
chance E2 0.6
chance E3 0.5
$offput
putclose emp;
```

Observe that every constraint is listed separately with its respective probability. Note that in case one constraint has to be satisfied in all scenarios (so it is strictly speaking not a chance constraint), then it has to be listed with probability 1.0.

Here is a summary of which constraints are satisfied in which scenarios in the optimal solution:

$$
\begin{array}{ll}
\text{Scenarios where E1 is satisfied:} & 4,5,6,7,8,9,10,11,12 \\
\text{Scenarios where E2 is satisfied:} & 2,3,5,6,8,9,11,12 \\
\text{Scenarios where E3 is satisfied:} & 6,7,8,9,10,11,12.
\end{array}
\tag{4.30}
$$

Observe that all constraints are satisfied in as many scenarios as required. Note that as predicted there are scenarios where all three constraints are satisfied ($k = 6,8,9,11,12$), scenarios where only two constraints are satisfied ($k = 5,7,10$), scenarios where only one constraint is satisfied ($k = 2,3,4$) and one scenario where all constraints are violated ($k = 1$).

4.4 Joint chance constraints vs. individual chance constraints

The choice whether joint or individual chance constraints should be used depends on the system being modeled. Both approaches have their own advantages. Individual chance constraints are weaker since not all constraints have to be satisfied at the same time. This can be clearly observed in the optimal solution for example (4.22). The objective value is 5.20 in the model with joint chance constraints and 4.75 in the model with individual chance constraints (assuming that each constraint is satisfied in 60% of all scenarios). Since it is a minimizing problem the model with individual chance constraints yields the better result. However, in this solution we have only 6 scenarios where both constraints are simultaneously satisfied while each constraint is satisfied in eight scenarios in total, as required.

4.5 Penalizing violations of chance constraints

EMP SP offers the syntax for a penalty factor for each scenario that violates one or more constraints. Taking the joint chance constraints example (4.22) the emp.info file of the GAMS code could be modified in the following way:

```
1 file emp / '%emp.info%' /; put emp '* problem %gams.i%'/;
2 $onput
3 randvar om1 discrete 0.25 1  0.25 2  0.25 3  0.25 4
4 randvar om2 discrete 0.3333 1  0.3334 2  0.3333 3
5 chance E1 E2 0.6 3
6 $offput
7 putclose emp;
```

Note that in line 5 we added the penalty factor 3. Recall the MIP equivalent (4.25) of the problem:

$$\begin{array}{ll} \text{Min} & z = x_1 + x_2 \\ \text{s.t.} & 1x_1 + x_2 \geq 7 - M(1-y_1) \\ & 1x_1 + 3x_2 \geq 12 - M(1-y_1) \\ & \vdots \\ & 4x_1 + x_2 \geq 7 - M(1-y_{12}) \\ & 3x_1 + 3x_2 \geq 12 - M(1-y_{12}) \\ & cc_1 = 1 - \sum_k \pi^k y_k, \quad k = 1,\ldots,12, \quad \pi^k = \frac{1}{12} \\ & x_1, x_2 \geq 0 \\ & 0 \leq cc_1 \leq (1-0.6) \\ & y_k \in (0,1). \end{array} \tag{4.31}$$

The probability with which the constraints were violated is stored in the variable cc_1 . The introduction of the penalty factor on line 5 of the emp.info file causes cc_1 multiplied by the penalty factor to be added to the objective function:

$$z = x_1 + x_2 + 3cc_1. \tag{4.32}$$

Similarly, the lines

```
chance E1 0.75 5
chance E2 0.6 6
chance E3 0.5 7
```

in the emp.info file of the GAMS code for the problem with individual chance constraints (4.26) trigger the objective function in the MIP equivalent to become:

$$z = x_1 + x_2 + 5cc_1 + 6cc_2 + 7cc_3. \tag{4.33}$$

Scenario	Probability	ATT	GMC	USX
s1	$\frac{1}{12}$	1.300	1.225	1.149
s2	$\frac{1}{12}$	1.103	1.290	1.260
s3	$\frac{1}{12}$	1.216	1.216	1.419
s4	$\frac{1}{12}$	0.954	0.728	0.922
s5	$\frac{1}{12}$	0.929	1.144	1.169
s6	$\frac{1}{12}$	1.056	1.107	0.965
s7	$\frac{1}{12}$	1.038	1.321	1.133
s8	$\frac{1}{12}$	1.089	1.305	1.732
s9	$\frac{1}{12}$	1.090	1.195	1.021
s10	$\frac{1}{12}$	1.083	1.390	1.131
s11	$\frac{1}{12}$	1.035	0.928	1.006
s12	$\frac{1}{12}$	1.176	1.715	1.908

Table 31.3: Return by scenario

This can be useful in order to explore sensitivities to slight changes.

There is also a possibility that allows the modeler to use the probability expression as a variable in the original model. For example, in the joint chance constraints problem above we could introduce a new variable, *viol*, in the objective function:

$$\text{Min} \quad z = x_1 + x_2 + 3 * viol \tag{4.34}$$

Then we replace line 5 in the emp.info file of the GAMS code as follows:

```
chance E1 E2 0.6 viol
```

This addition causes cc_1 to be replaced by *viol* in the MIP equivalent. Thus we have:

$$viol = 1 - \sum_k \pi^k y_k, \quad k = 1, \ldots, 12, \quad \pi^k = \frac{1}{12}, \quad viol \in [0, 0.4]. \tag{4.35}$$

This model is equivalent to the joint chance constraints model with penalty factor 3 with which we started this section.

5 Risk Measures

Risk measures are mechanisms to evaluate the effects of uncertainty in the underlying system on the outcomes of interest. They can be used to modify the distribution of outcomes. In this section we explore how optimization problems involving risk measures can be modeled using stochastic programming in GAMS EMP. Specifically, we examine how an investor might seek to balance expected rewards and the risk of loss when she decides how to allocate assets in a portfolio.

First, we will discuss maximizing the expected value of a portfolio with uncertain returns, then we will briefly introduce the notion of Value at Risk and finally we will examine optimization problems involving Conditional Value at Risk and a combination of expected value and Conditional Value at Risk. For simplicity we consider only one period of uncertainty $(0,T)$ between the two dates 0 and T. In the examples introduced below the period $(0,T)$ is the period between investing in a portfolio of assets and return from this portfolio.

5.1 Expected Value

Suppose an investor has the opportunity to invest a certain amount in three assets. She is given the probability distribution in Table 31.3 that links each asset with a possible return at time T. The question arises how she should allocate her funds between the three assets at time 0 in order to maximise her *expected* return at time T.

Mathematically, the problem can be expressed as follows:

$$\begin{array}{ll} \text{Max} & \mathbb{E}[R] \\ \text{s.t} & R = \sum_j w_j v_j \\ & \sum_j w_j = 1 \\ & w_j \geq 0, \end{array} \tag{5.36}$$

where the variable R is the return (and is a function of the random variable v), $\mathbb{E}[R]$ the expected return, w_j the weight associated with each asset j and v_j is the (random variable) return of each asset j. The weights can also be interpreted as proportions of the amount to be invested, their sum must be 1. Note that the w_j's are the decision variables in this problem.

We present two different ways to model this problem in GAMS using EMP SP. Both models have two stages: in the first stage the weights are chosen without knowing which scenario will be realized, in the second stage the 12 scenarios are taken into account. We start with the part of the code where the data is given. It is named `data.inc` and incorporated in all models in this section.

```
Set j assets    / ATT, GMC, USX /
    s scenarios / s1*s12 /

Table vs(s,j) scenario returns from assets
     att     gmc     usx
 s1  1.300   1.225   1.149
 s2  1.103   1.290   1.260
 s3  1.216   1.216   1.419
 s4  0.954   0.728   0.922
 s5  0.929   1.144   1.169
 s6  1.056   1.107   0.965
 s7  1.038   1.321   1.133
 s8  1.089   1.305   1.732
 s9  1.090   1.195   1.021
s10  1.083   1.390   1.131
s11  1.035   0.928   1.006
s12  1.176   1.715   1.908;

Alias (j,jj);
Parameter
    mean(j)      mean return
    dev(s,j)     deviations
    covar(j,jj)  covariance matrix of returns
    totmean      total mean return;

mean(j)     = sum(s, vs(s,j))/card(s);
dev(s,j)    = vs(s,j) - mean(j);
covar(j,jj) = sum(s, dev(s,j)*dev(s,jj))/(card(s)-1);
totmean     = sum(j, mean(j))/card(j);
display mean, dev, covar, totmean;

Parameter
    p(s)   probability / #s [1/card(s)] /
    v(j)   return from assets; v(j) = mean(j);
```

Here is the first model:

```
$include data.inc

Variables
        r      value of portfolio under each scenario
        w(j)   portfolio selection;
Positive variables w;

Equations
```

```
9           defr     return of portfolio
10          budget   budget constraint;
11
12  defr..      r =e= sum(j, v(j)*w(j));
13  budget..    sum(j, w(j)) =e= 1;
14
15  model portfolio / all /;
16
17  Parameter
18      s_v(s,j)     return from assets by scenario /s1.att 1/
19      s_r(s)       return from portfolio by scenario;
20
21  Set dict / s     .scenario.''
22             v     .randvar. s_v
23             r     .level.   s_r /;
24
25  file emp / '%emp.info%' /; emp.nd=4;
26  put emp '* problem %gams.i%'
27       / 'stage 2 r defr v'
28       / 'jrandvar v('att') v('gmc') v('usx')'
29  loop(s,   put / p(s) vs(s,"att") vs(s,"gmc") vs(s,"usx"));
30  putclose emp;
31
32  solve portfolio using emp max r scenario dict;
33  display s_r, w.l;
```

As usual, an emp.info file is created (lines 25-30) and the set `dict` is defined (lines 21-23) for output handling. Observe that the syntax suggests that the return `r` is maximized. (line 32). However, in fact the *expected* return is maximized. Note that the statement `emp.nd=4` ensures that 4 decimal places are used for the values of `vs` in the emp file, the default is 2. Note further that the solver renormalizes the sum of the probabilities to 1 if some input rounding is performed.

In the second model it is explicitly stated that the *expected value* of the return is maximized. We introduce a new variable for the expected return, `EV_r`, and the new keword `ExpectedValue`.

```
1   $include data.inc
2
3   Variables
4       r         value of portfolio under each scenario
5       w(j)      portfolio selection
6       EV_r      expected value of r
7       objective objective variable;
8   Positive variables w;
9
10  Equations
11      defr     return of portfolio
12      budget   budget constraint
13      obj_eq         objective eqn;
14
15  defr..      r =e= sum(j, v(j)*w(j));
16  budget..    sum(j, w(j)) =e= 1;
17  obj_eq..    objective =e= EV_r;
18  model portfolio / all /;
19
20  file emp / '%emp.info%' /;
21  emp.nd=4;
22  put emp '* problem %gams.i%'
23          / 'ExpectedValue r EV_r'
24        / 'stage 2 v defr r'
25        / 'stage 1 objective obj_eq EV_r'
26        / "jrandvar v('att') v('gmc') v('usx')"
27  loop(s,
```

```
  put / p(s) vs(s,"att") vs(s,"gmc") vs(s,"usx"));
putclose emp;

Parameter
    s_v(s,j)     return from assets by scenario /s1.att 1/
    s_r(s)       return from portfolio by scenario;

Set dict / s     .scenario.''
           v     .randvar. s_v
           r     .level.   s_r /;

solve portfolio using emp max objective scenario dict;
display s_r, w.l;
```

In this model the objective to be maximized is `EV_r`, the expected return. In the emp file `EV_r` is declared as the `ExpectedValue` of the random variable `r`. Note that the variables and the equation of stage 2 remain unchanged, but the new variables `EV_r` and `obj` belong to stage 1. Since the expected value of `r` is not scenario dependent, its value is known in the preceding stage to the resolution of `r`, namely stage 1. Note that in a 3-stage-problem with `r` in the third stage, the expected value of `r` will be known with certainty in the second stage. Both models have the same solution. We prefer the second model since the syntax is more explicit and clearer.

5.2 Value at Risk (VaR)

Suppose $G(x,\xi)$ is a real valued function of the decision vector x and a random data vector ξ and that it denotes the *loss* function of a portfolio of assets. We aim to restrict potential losses and so we choose a portfolio composition such that the loss does exceed a certain threshold γ ($\gamma \in \mathbb{R}$) with a probability smaller or equal to α, $\alpha \in (0,1)$, where α is small. This condition can be modeled as a chance constraint (compare section 4 on chance constraints) and has the form

$$P(G(x,\xi) > \gamma) \leq \alpha \tag{5.37}$$

or equivalently,

$$P(G(x,\xi) - \gamma \leq 0) \geq 1 - \alpha. \tag{5.38}$$

Consider the random variable $Z_x := G(x,\xi) - \gamma$. For a given value of x, let $F_Z(z) := P(Z \leq z)$ be the cumulative distribution function of Z. Now, the point x satisfies the constraint (5.38) if and only if $F_Z(0) \geq 1-\alpha$. This is equivalent to saying that x satisfies the constraint (5.38) if and only if $F_Z^{-1}(1-\alpha) \leq 0$.

The (left-side) quantile $F_Z^{-1}(\theta)$ is called *Value at Risk*. It is denoted by $VaR_\theta(Z)$, i.e.

$$VaR_\theta(Z) = \inf\{t : F_Z(t) \geq \theta\}. \tag{5.39}$$

Hence constraint (5.38) can be written in the following equivalent form:

$$VaR_{1-\alpha}(G(x,\xi)) \leq \gamma. \tag{5.40}$$

The figure below illustrates $VaR_{0.05}(Z)$ where Z is normally distributed with mean $\mu = 0.645$ and standard deviation $\sigma = 1$.

5.3 Conditional Value at Risk (CVaR)

$\underline{CVaR}_\alpha$ is the expected average return (in a given time period) given that we are in the $(\alpha \times 100)\%$ *left tail* of the return distribution, where $\alpha \in (0,1)$. In other words, $\underline{CVaR}_\alpha$ is a mean of the left tail. For example, if we are interested in the 5% worst cases, i.e. $\alpha = 0.05$, $\underline{CVaR}_\alpha$ is the conditional expection of the return, given the return is no greater than VaR.

Let ξ be a random variable with probability density function $p(\xi)$, let $G(\xi)$ be a function of the random variable ξ denoting the *return* of a portfolio of assets and let $\alpha \in (0,1)$ be a probability. Then the conditional value at risk of $G(\xi)$ is defined as

$$\underline{CVaR}_\alpha(G(\xi)) = \frac{1}{\alpha}\int_{-\infty}^{VaR_\alpha} G(\xi)\cdot p(\xi)d\xi, \tag{5.41}$$

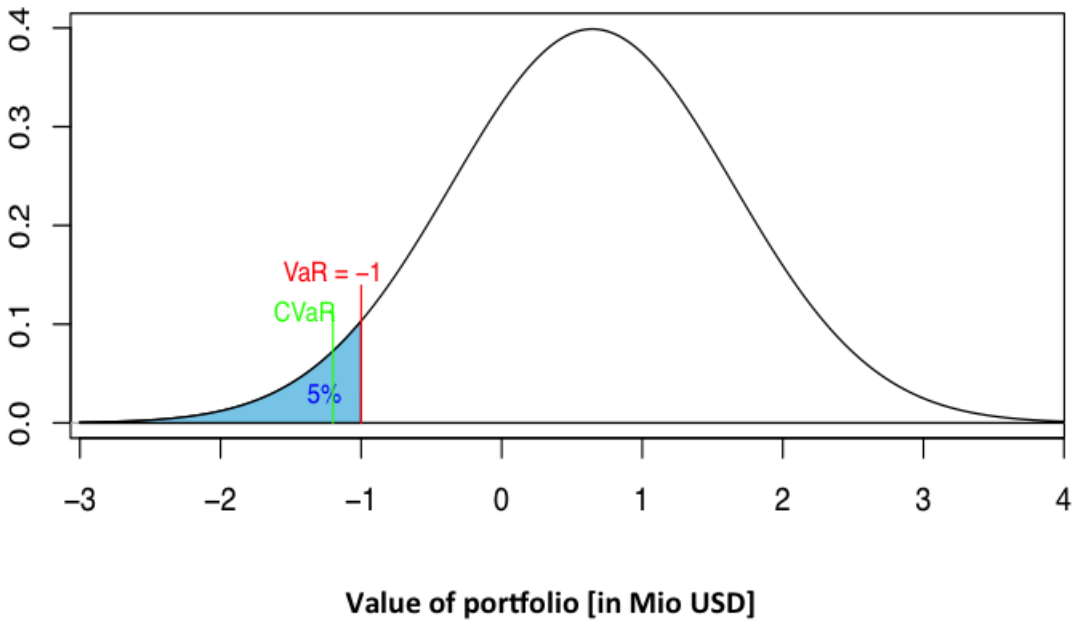

Figure 31.2: VaR and CVaR

where VaR_α is the value at risk.

In the example above the investor might be interested to make sure that if things get bad she loses as little as possible. She might consider the worst 10% of possible cases and allocate her funds such that the expected mean return in these cases is maximized. Mathematically, the problem can be expressed as follows:

$$\begin{array}{ll} \text{Max} & \underline{CVaR}_\theta[R] \\ \text{s.t} & R = \sum_j w_j v_j \\ & \sum_j w_j = 1 \\ & w_j \geq 0, \end{array} \tag{5.42}$$

where $\mathbb{E}[R]$ is the expected value of the return and $\underline{CVaR}_\theta$ is the CVaR at the confidence level θ.

This problem can be modeled in GAMS by introducing in the emp file a new variable for the conditional value at risk (`CVar_r`), a scalar to specify the percentage of the worst cases we are interested in (`theta`) and `cvarlo`, a new keyword.

```
$include data.inc

Scalar
    theta          relative volume / [1-0.9] /;

Variables
    r                value of portfolio under each scenario
    w(j)             portfolio selection
    CVaR_r    conditional value at risk of r
    objective objective variable;
Positive variables w;
```

```
12
13 Equations
14     defr    return of portfolio
15     budget  budget constraint
16     obj_eq  objective function;
17
18 defr..     r =e= sum(j, v(j)*w(j));
19 budget..   sum(j, w(j)) =e= 1;
20 obj_eq..   objective =e= CVaR_r;
21 model portfolio / all /;
22
23 file emp / '%emp.info%' /
24 put emp '* problem %gams.i%'
25       / 'cvarlo r CVaR_r ' theta
26       / 'stage 2 r defr v'
27       / 'stage 1 objective obj_eq CVaR_r'
28       / "jrandvar v('att') v('gmc') v('usx')"
29 loop(s,
30   put / p(s) vs(s,"att") vs(s,"gmc") vs(s,"usx"));
31 putclose emp;
32
33 Parameter
34     s_v(s,j)     return from assets by scenario /s1.att 1/
35     s_r(s)       return from portfolio by scenario;
36
37 Set dict / s     .scenario.''
38            v     .randvar. s_v
39            r     .level.   s_r /;
40
41 solve portfolio using emp max objective scenario dict;
42 display s_r;
```

Observe that the objective equals the Conditional Value at Risk (the conditional expectation of the *left* tail, denoted by $\underline{CVaR}$). Line 25 specifies that the variable `CVaR_r` is the Conditional Value at Risk, `r` is the random variable and the scalar `theta` is the fraction (in range 0 to 1) we consider. As in the previous section, the objective, the objective equation and the variable of the objective belong to the first stage while the the equation that handles the random data and all its variables belong to the second stage.

Note that the keyword `cvarlo` specifies that we are looking at the *left* tail of the probability distribution. For the *right* tail of the distribution the keyword to be used is `cvarup`. The keyword `cvar` is identical to `cvarup`. The conditional value of risk denoting the mean of the *right* tail of the distribution can be denoted by $\overline{CVaR}$ and is defined as:

$$\overline{CVaR}_\alpha(G(\xi)) = \frac{1}{1-\alpha}\int_{VaR_\alpha}^{\infty} G(\xi)\cdot p(\xi)d\xi. \tag{5.43}$$

Note that it is only appropriate to maximize $\underline{CVaR}_\alpha$ and minimize $\overline{CVaR_\alpha}$. Furthermore, $\underline{CVaR}_\alpha$ is a concave function and so should only be constrained using e.g.

$$\underline{CVaR}_\alpha \geq \gamma$$

and $\overline{CVaR_\alpha}$ is convex, so should only appear in constraints like

$$\overline{CVaR_\alpha} \leq \gamma.$$

In a final variation on the example above we consider an investor who aims to take into account both, the expected return and the Conditional Value at Risk of the return at a certain threshold θ. She combines the two risk measures and uses a scalar (λ) to weigh the two summands. A mathematical formulation of the problem reads as follows:

$$\begin{array}{ll} \text{Max} & \lambda\mathbb{E}[R] + (1-\lambda)CVaR_\theta[R] \\ \text{s.t} & R = \sum_j w_j v_j \\ & \sum_j w_j = 1 \\ & w_j \geq 0, \end{array} \tag{5.44}$$

where $\mathbb{E}[R]$ is the expected value of the return and $CVaR_\theta$ is the CVaR at the confidence level θ.

The GAMS model follows.

```
$include data.inc

Scalar
    theta  relative volume / [1-0.9] /
    lambda weight EV versus CVaR / 0.2 /;

Variables
    r       value of portfolio under each scenario
    w(j)    portfolio selection
    CVaR_r  conditional value at risk of r
    EV_r    expected value of r
    obj     objective variable;
Positive variables w;

Equations
    defr    return of portfolio
    budget  budget constraint
    defobj  convex combination for both risk measures;

defr..      r =e= sum(j, v(j)*w(j));
budget..    sum(j, w(j)) =e= 1;
defobj..    obj =e= lambda*EV_r + (1-lambda)*CVaR_r;
model portfolio_ext / all /;

file emp / '%emp.info%' /
put emp '* problem %gams.i%'
      / 'ExpectedValue r EV_r'
      / 'cvarlo r CVaR_r ' theta
      / 'stage 2 r defr v'
      / 'stage 1 defobj obj'
      / "jrandvar v('att') v('gmc') v('usx')"
loop(s,
  put / p(s) vs(s,"att") vs(s,"gmc") vs(s,"usx"));
putclose emp;

Set dict / s     .scenario.''
           v     .randvar. s_v
           r     .level.   s_r /;

solve portfolio_ext using emp max obj scenario dict;
display s_r;
display CVaR_r.l;
display EV_r.l;
```

The scalar `lambda` is introduced as a weight in the sum in the objective function and in the emp file both keywords `ExpectedValue` and `cvarlo` are used. Other risk measures could be implemented in future versions of EMP SP.

Concluding this section we present an alternative way to model CVaR. The code is identical to the code of the first model on page 614 except for the scalar `theta` and the emp.info file. The modification of the code for this alternative way of modeling CVaR is given below.

```
Scalar
    theta  relative volume / [1-0.9] /;

file emp / '%emp.info%' /
put emp '* problem %gams.i%'
      / 'cvarlo ' theta
```

```
      / 'stage 2 r defr v'
      / "jrandvar v('att') v('gmc') v('usx')"
loop(s,   put / p(s) vs(s,"att") vs(s,"gmc") vs(s,"usx"));
putclose emp;

solve portfolio using emp max r scenario dict;
display s_r;
```

Observe that there is no additional variable for CVaR, but the risk measure is simply applied to the objective function.

6 Summary of keywords and solver configurations

The following keywords can be used in the emp.info file to describe the uncertainty of a problem:

chance: This defines individual or joint chance constraints (CC) using the following syntax:

`chance equ {equ} [holds] minRatio [weight|varName]`

This way one defines that a single constraint `equ` (individual CC) or a set of constraints (joint CC) does only have to hold for a certain ratio ($0 \leq$ `minRatio` ≤ 1) of the possible outcomes. The keyword `holds` is optional and does not affect the solver. If `weight` is defined, the violation of a CC gets penalized in the objective (`weight` $*$ `violationRatio`). Alternatively, the violation can be multiplied by an existing variable if this is defined by `varName`.

cvarlo: This keyword assigns a variable to have the value $\underline{CVaR}_\alpha$. α is a scalar that represents the confidence level for the Conditional Value at Risk. There are two options for the syntax:

`cvarlo scalar`

and

`cvarlo rv var scalar`

In the first option the objective function variable is used, whereas in the second option the random variable used is named explicitly (rv) and a variable for the value of CVaR is added (var), and scalar is the value of α.

cvarup: This keyword assigns a variable to have the value $\overline{CVaR}_\alpha$. α is a scalar that represents the confidence level for the Conditional Value at Risk. There are two options for the syntax:

`cvarup scalar`

and

`cvarup rv var scalar`

In the first option the objective function variable is used, whereas in the second option the random variable used is named explicitly (rv) and a variable for the value of CVaR is added (var), and scalar is the value of α. Note that the keyword `cvar` is identical to `cvarup` which refers to the right tail of the distribution.

ExpectedValue: This keyword is used to state that a variable is the expected value of a random variable. The syntax is as follows:

`ExpectedValue rv var`

jrandvar: `Jrandvar` can be used to define discrete random variables and their joint distribution:

`jrandvar rv rv {rv} prob val val {val} {prob val val {val}}`

At least two random variables `rv` are defined and the outcome of those is coupled. The probability of the outcomes is defined by `prob` and the corresponding realization for each random variable by `val`.

randvar: This defines both discrete and parametric random variables:

```
randvar rv discrete prob val {prob val}
```

The distribution of discrete random variables is defined by pairs of the probability `prob` of an outcome and the corresponding realization `val`.

```
randvar rv distr par {par}
```

A list of all supported parametric distributions can be found in Table 31.2 on page 599. All possible values for `distr` and the related parameters `par` are listed there.

sample: This allows the user to customize the size of the sample of a random variable from a continuous distribution and there is also the option to determine the variance reduction method to be used:

```
sample  rv1 [rv2 ... rvn] sampleSize [varRedMethod]
```

In `rv` the name of the random variable is given. The sample size of more than one random variable can be customized simultaneously. In this case the names of the random variables are listed. `sampleSize` is a number, namely the desired size of the sample and `varRedGroup` is optional. For more details about variance reduction methods see section 2.3 and in addition, please consult the LINDO manual.

setSeed: This sets the seed for the random number generator of the sampling routines called using the `sample` keyword. If `setSeed` is used in the emp.info file, the seed is set once before we generate all samples.

```
setSeed seed
```

stage: Random variables (`rv`), equations (`equ`) and variables (`var`) are assigned to non-default stages like this:

```
stage stageNo rv | equ | var {rv | equ | var}
```

`StageNo` defines the stage number. The default stage for all random variables, equations and variables not mentioned with the `stage` keyword is 1, except for the objective equation. The default for the objective equation is the highest stage mentioned. Note that the objective variable is in stage 1.

At the moment four GAMS solvers can be used to solve SP models in the way descibed in this document: DE, DECIS, LINDO and JAMS. Further information about these solvers can be found in the corresponding solver manuals.

Not all keywords mentioned above are supported by all four solvers. The following table specifies which keywords can be used with which solvers. The keywords not mentioned in the table are supported by all four solvers.

	DE	DECIS	LINDO	JAMS
chance	√		√	√
jrandvar	√	√	√	
randvar (discrete)	√	√	√	
randvar (parametric)			√	
sample			√	
setSeed			√	
cvar	√			√
ExpectedValue	√			√

Table 31.4: Solver Capabilities

The SP options available for the LINDO solver are documented in the LINDO/LINDOGlobal manual.

7 More on scenarios and output extraction

The size of the set `scen` does not have to match the number of scenarios actually generated in the solution process. For example, if we set the size of `scen` to 2 in the news vendor model from section 2.1, then the results of the first two scenarios will be stored in the parameters `s_d, s_x` and `s_s`, and the results of the other scenarios will not be stored. On the other

hand, if the size of scen is bigger than the number of scenarios generated, then in the parameters (e.g. s_d) the positions of the surplus elements of scen will be empty.

After solving a SP model only the solution of the expected value problem can be accessed via the regular .L and .M fields. As described in section 2.1, additional parameters have to be defined to store the results for the different scenarios solved. The following values can be stored by this approach:

level:	Stores the levels of a scenario solution of variable or equation.
marginal:	Stores the marginals of a scenario solution of variable or equation.
randvar:	Stores the realization of a random variable.
opt:	Stores the probability of each scenario.

In the news vendor model from section 2.1 we can use the following:

```
Set scen        Scenarios / s1*s6 /;
Parameter
  s_x(scen)        Units bought by scenario
  s_rep(secen,*)   Scenario probability         / #scen.prob 0/;

Set dict / scen .scenario.''
           x    .level   .s_x
           ''   . opt    .s_rep/;
```

The size of the set scen defines the number of scenarios we are willing to store results for. X is a variable for which we want to access the level and s_x is the parameter the levels of x are stored in. Note that s_x needs to have the same indices as x plus the additional index scen in the first position. In the parameter s_rep we store the probabilities of the different scenarios solved.