University of Vienna
Institute of Statistic and Decision Support Systems

A Tool for Modeling Asset Management Problem

Mariusz Siomak

The work described in this report was supported
by the Special Research Program SFB F011 "AURORA"
of the Austrian Science Fund.

VIENNA 2000

Contents

Chapter 1

Introduction

This document presents the issue of asset management modeling and describes a tool designed for generating such models. Models considered here belong to a very general class of decision models characterized by the following assumptions:

1. the decision problem is a discrete-time multi-period decision problem over finite horizon
2. the decision problem involves uncertainty that is expressed as multivariate stochastic forecast of some model parameters (stochastic model)
3. the decision problem is linear with respect to decision variables (linear optimization problem)

The above assumptions imply that the model is a multi-period linear stochastic program. Nonetheless such a model can be represented in different forms suitable for specific solving methods.

The most general method of representing a linear program is in the standard form (a form in which constraints are expressed as $Ax = b$; $x \geq 0$). Convenient interface for linear program in its standard form is an MPS file. Dynamic stochastic linear program can be given in the standard form by converting it to so-called deterministic equivalent (see [KW94]). Such formulation is suitable for any general LP solver. The drawback is that the matrix A is huge and usually very sparse (which on the other hand may be explored by some solvers), and the structure of the problem is lost.

The other method of representing a multi-period linear stochastic program is to save its structure and data together. The example of the interface for such formulation is described in [BDG$^+$87]. This form is suitable for solvers taking into account the structure of the problem, usually for any decomposition methods (e.g., nested Benders decomposition).

For the model to be generated two kinds of information are necessary: the description of the multi period decision problem and the description of multivariate stochastic forecast. They are given in a model description file and a tree file. This structure is shown in the gray box in Fig. 1.1 (as a part of the whole Aurora DSS system).

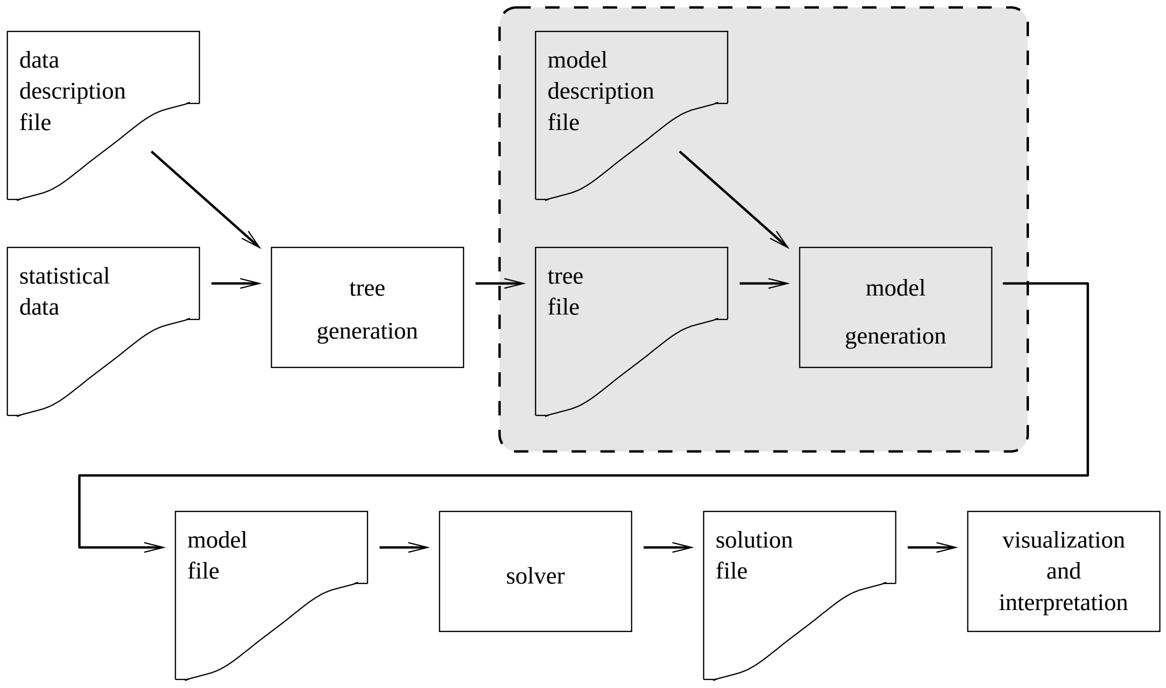

Figure 1.1: The Aurora decision Support System (part described in this document is in gray box).

Chapter 2

Asset liability management model

2.1 Modeling of the financial instruments

Let us assume that there are J financial instruments (and they are assigned numbers $1, \dots, J$). The control horizon consists of time instants denoted by numbers $0, 1 \dots, I$ (so there are $I + 1$ time stages on the horizon). At stages $0, 1, \dots, I$ decisions are made how many instruments of each type buy or sell. At stage I we can observe the final results of past decisions. With each financial instrument four parameters are associated:

n^s_{ij} - sell price of j-th instrument at stage $i \in \{0, \dots, I\}$ (a price for which the contract may be sold)

n^b_{ij} - buy price of j-th instrument at stage $i \in \{0, \dots, I\}$ (a price for which the contract may be bought)

n^f_{ij} - cash flow of j-th instrument at stage $i \in \{0, \dots, I\}$ (resulting cash flow, for example dividends, coupon payments)

n^v_{ij} - value of j-th instrument at stage $i \in \{0, \dots, I\}$ (a value needed to calculate wealth at each node, or at least at the terminal nodes)

With each contract three non-negative decision variables are connected:

x^s_{ij} - a number of instruments of the type j to be sold at stage $i \in \{0, \dots, I\}$

x^b_{ij} - a number of instruments of the type j to be bought at stage $i \in \{0, \dots, I\}$

x^n_{ij} - a number of instruments of the type j hold at stage $i \in \{0, \dots, I\}$ (just after decisions have been made)

Other variables in the model are connected with time stages. They are:

x^c_i - cash hold at time stage $i \in \{0, \dots, I\}$ (just after decisions have been made)

x^w_i - total wealth of the portfolio at time stage $i \in \{0, \dots, I\}$ (just after decisions have been made)

$x^{\Delta w}_i$ - wealth increase at time stage $i \in \{0, \dots, I\}$ (just after decisions have been made)

Some variables have their initial values. These are:

$x^n_{-1,j}$ - a number of instruments of the type j hold just before time 0

x^c_{-1} - cash hold just before time 0

There are also two parameters describing internal cache flows:

c^i_k - external cache inflow at time stage $k \in \{0, \dots, I\}$

c_k^o - external cache outflow at time stage $k \in \{0, \ldots, I\}$

A multistage asset management model is described by the following bookkeeping equations:

$$x_i^c = x_{i-1}^c + \sum_{j=1}^{J} \left(n_{ij}^s x_{ij}^s - n_{ij}^b x_{ij}^b + n_{ij}^f x_{ij}^n \right) + c_i^i - c_i^o; \quad i = 0, \ldots, I \tag{2.1}$$

$$x_{ij}^n = x_{i-1,j}^n + x_{ij}^b - x_{ij}^s; \quad i = 0, \ldots, I, j = 1, \ldots, J \tag{2.2}$$

$$x_i^w = x_i^c + \sum_{j=1}^{J} \left(n_{ij}^v x_{ij}^n \right); \quad i = 0, \ldots, I \tag{2.3}$$

$$x_i^{\Delta w} = x_i^w - x_{i-1}^w; \quad i = 1, \ldots, I \tag{2.4}$$

together with inequalities: $x_i^c \geq 0$, $x_{ij}^n \geq 0$ and $x_i^w \geq 0$.

Note 1 The variable $x_i^{\Delta w}$ can be of any sign, so it is split in the model into a positive and a negative part: $x_i^{\Delta w} = x_i^{\Delta w(+)} - x_i^{\Delta w(-)}$, where $x_i^{\Delta w(+)} \geq 0$ and $x_i^{\Delta w(-)} \geq 0$.

Note 2 The variable $x_0^{\Delta w}$ is in fact incorporated into the model and the user has access to it by predefined variable `WEALTH_INCREASE[0]`, but it must be stressed, that this value is reliable solely when there are no initial contract quantities or the value of each contract that is given some initial quantity can be calculated for time index -1 (only in these cases the initial wealth can be properly calculated).

2.2 Modeling of risk

Risk is incorporated into a model by means of stochastic forecast expressed by a tree. This tree shows possible scenarios. Each node is assigned probability of attaining it $p(n)$. In each node of the tree there could be some values of model parameters. Nodes are numbered from 1 to N (node 1, that is root, is assigned to time index 0). Leaves of this tree are called *terminal nodes.* In the example shown in Fig. 2.1 terminal nodes are nodes with numbers 4,5,6,7,8, and 9.

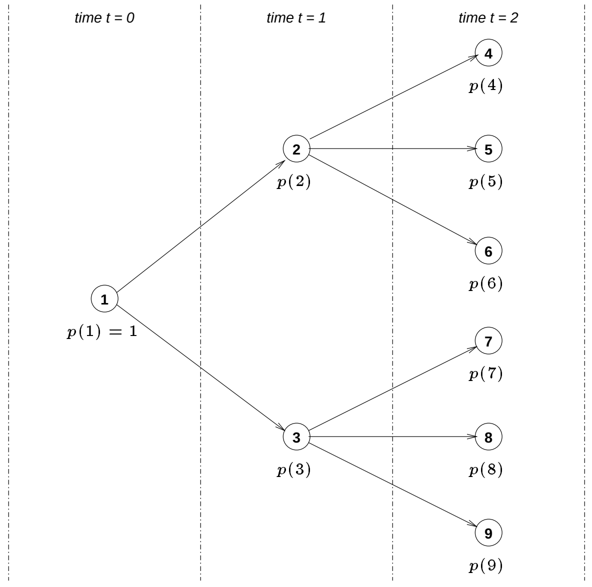

Figure 2.1: Multistage probabilistic tree structured forecast

2.3 Modeling of the objective function

There are two criteria for objective function: maximizing expected terminal wealth, and minimizing the risk. They are incorporated into the model.

Terminal wealth is a random variable since it depends on scenario. At each terminal node n of the scenario tree, the wealth variable $W(n)$ is known together with the probability $p(n)$ that this node is reached.

Risk is measured by mean absolute deviation (MAD):

$$\mathrm{E}\left(\mid W - \mathrm{E}(W) \mid\right). \tag{2.5}$$

The whole performance function is thus of the form:

$$\max\ \mathrm{E}(W) - \rho \mathrm{E}\left(\mid W - \mathrm{E}(W) \mid\right) \tag{2.6}$$

where $0 < \rho \leq \frac{1}{2}$ measures the degree of risk aversion.

The objective function is incorporated into the linear model in the following manner. The objective of the linear program becomes:

$$\max\ s_1 - \rho \sum_{n \in T} p(n)\left(s_2(n) + s_3(n)\right) \tag{2.7}$$

where:

$$s_1 = \sum_{n \in T} p(n) x_I^w(n) \tag{2.8}$$

$$x_I^w(n) - s_1 = s_2(n) - s_3(n) \quad n \in T \tag{2.9}$$

and T is a set of indices of terminal nodes.

Chapter 3

Model Description File

3.1 The Syntax of a Model Description File

The file describing decision problem consists of sections defining particular aspects of a model, such as scenario tree, contracts, constraints,initial values as well as goals. It also contains a section for defining some parameters used in the model. The general structure of this file is shown below.

```
PROBLEM name
TYPE PORTFOLIO OPTIMIZATION

PARAMETERS
    parameter section
END PARAMETERS

TREE
    tree section
END TREE

CONTRACTS
    contract list
END CONTRACTS

CONTRACT name
    contract section
END CONTRACT

CONSTRAINTS
    constraints section
END CONTSTRAINTS

INITIAL VALUES
    initial values section
END INITIAL VALUES

EXTERNAL FLOWS
    external flows section
```

```
END EXTERNAL FLOWS

GOAL
```
goal section
```
END GOAL

END PROBLEM
```

3.2 Objects used in description file

Definition of a mathematical programming problem requires us to specify its variables and some relationships between them. The purpose of description file is to support a user in defining an optimization problem for optimal asset allocation under stochastic, multivariate forecast.

Two main classes of objects used in description file are: parameters and variables. Parameters reflect values, that remain constant during the problem solution, are not subject of optimization. Variables, on the other hand, correspond to values that are subject of optimization. Parameters and variables may be scalars or arrays (vectors of values).

There are three types of parameters as regards their source: *predefined parameters*, *user defined parameters*, and *label parameters.*

Now there is only one predefined parameter in an application. It is shown in Table 3.1.

name	meaning
`N`	scalar parameter, the time stage considered; to be used only in contract and constraint sections

Table 3.1: Predefined parameter

User defined parameters are used to substitute constant values by meaningful names. Their use is sometimes forced by the file syntax, for example past realizations of values defined by a tree, or external flows may be given only by array parameters.

Label parameters are defined by the application during parsing TREE SECTION. They denote the sequence of values (realization) of a stochastic variable on the whole horizon.

Except for parameters, some variables are also defined when a model is created. The use of variables is the same as of parameters, the only difference is that a user cannot define variables themselves. Variables represent values that are subject of optimization and are not fixed.

There are two types of variables as regard their origin: *contract variables*, and *predefined variables.*

Contract variables are defined when parsing CONTRACTS SECTION. They reflect the number of contracts and their value. Predefined variables are shown in Table 3.2. They help to define and solve the problem. Variables can be used only in `CONSTRAINTS` and `INITIAL VALUES` sections.

3.3 Parameter Section

A parameter section introduces parameters that will be used in subsequent sections to represent constant data. There are two types of parameters: scalar parameter and array parameter. A scalar parameter defines a single value whereas an array parameter is used for storing a one-dimensional array of values.

In general, a parameter section has the following structure:

`PARAMETERS`

scalar parameter definition | *array parameter definition*

name	meaning
CASH	array variable, denotes the amount of cash for each time stage
WEALTH	array variable, denotes the value of the portfolio for each time stage
WEALTH_INCREASE	array variable, denotes the increase of the portfolio value for each time stage

Table 3.2: Predefined variables

Section	objects
Predefined	Predefined parameters and variables: N, CASH, WEALTH, WEALTH_INCREASE
PARAMETERS	user defined parameters
TREE	label parameters
CONTRACTS	contract variables
CONTRACT	
CONSTRAINTS	
INITIAL VALUES	
EXTERNAL FLOWS	
GOAL	

Table 3.3: Where objects are defined.

scalar parameter definition | *array parameter definition*
scalar parameter definition | *array parameter definition*
END PARAMETERS
where each *scalar parameter definition* is built as follows:
name = *expression*
and *array parameter definition* as follows:
name[*lower index*:*higher index*] = [*expression*, *expression*, ..., *expression*]

Lower index has to be less then or equal to *higher index*. Each of the expressions used to define scalar or array parameters has to be constant expression and may use parameters defined so far. For *array parameters* the number of expressions in square brackets has to be equal *higher index* - *lower index* + 1.

EXAMPLE 1

```
PARAMETERS
   NO_OF_CONTRACTS        = 5
   NO_OF_PERIODS          = 7
   COMMISSION             = 0.04
   J                      = 0.01
   PAST_VALUES[-5:-1]     = [1.1, 2.0, 1.0, 0.9, 0.8]
   MIC_PAST[-1:-1]        = [45.7]
   COEF[1:NO_OF_CONTRACTS] = [J, 5*J, 3*J, 2*J, 3*J-0.015]
END PARAMETERS
```

There is one mandatory parameter that has to be defined, because it is used internally by the application. This parameter is shown in Table 3.5.

Section	N	defined parameters	CASH, WEALTH, WEALTH_INCREASE	label parameters	contract variables
PARAMETERS		×			
TREE		×			
CONTRACTS					
CONTRACT	×	×		×	
CONSTRAINTS	×	×	×	×	×
INITIAL VALUES		×			×(*)
EXTERNAL FLOWS	×	×			
GOAL		×			

(*) In INITIAL VALUES section contract variables may be used only in left hand side

Table 3.4: Where objects can be used.

name	meaning
NO_OF_PERIODS	the number of time periods considered

Table 3.5: Mandatory parameter

3.4 Tree Section

The purpose of this section is to define a tree used in a model and to change some of its parameters. The general structure of a tree section is as follows:

```
TREE
    FILE NAME path
    LABEL label  | LABEL name [ tree label ]
        MEAN        = expression
        VARIANCE    = expression
        TREND       = expression
        PAST VALUES = array parameter
    LABEL label  | LABEL name [ tree label ]
        MEAN        = expression
        VARIANCE    = expression
        TREND       = expression
        PAST VALUES = array parameter
END TREE
```

The description of fields in a tree section is given in Table 3.6. The only mandatory field is FILE NAME so the simplest tree section is given by the Example 2.

EXAMPLE 2

```
TREE
   FILE NAME = treefile
END TREE
```

The field LABEL is used if there is a need to change some properties of values stored in the tree nodes. The syntax LABEL *label* [*tree label*] allows to change the label being an etiquette of a value. The name *label* in the model is equivalent to the label *tree label* in the tree.

name	meaning
`FILE NAME`	the name of a file containing a tree (mandatory)
`LABEL` *label*	the label of values given in tree nodes; it has to be defined in the tree file
`LABEL` *label* `[`*tree label*`]`	the label of values given in tree nodes; used to redefine label, in tree file label *tree label* has to be defined and it is known in a model as *label*
`MEAN`	used to change the mean value of a variable stored under label *label*
`VARIANCE`	used to change the variance of a variable stored under label *label*
`TREND`	used to change the trend of a variable stored under label *label*
`PAST VALUES`	used to set past realizations of a value stored under label *label*

Table 3.6: Fields of tree section

Such properties as mean value, variance and trend can be changed. The new mean value, trend and variance of a specific variable may be set only if corresponding old values are specified in the tree file. If not, an error will occur. If only some of the: mean, trend, variance is changed, the other remains unchanged. The values corresponding to the specified label are changed according to the formula (3.1) (t denotes time stage number starting from 0).

$$v_{new}(t) = MEAN_{new} + (v_{old}(t) - MEAN_{old} - t \cdot TR_{old})\sqrt{\frac{VAR_{new}}{VAR_{old}}} + t \cdot TR_{new} \tag{3.1}$$

Setting past realizations of a random variable represented by a tree is sometimes necessary because they may be used for correlated values (see Example 8, page 13). The argument assigned to the field `PAST VALUES` has to be an array parameter whose *lower index* and *higher index* are negative numbers.

EXAMPLE 3

```
TREE
   FILE NAME = tree1
   LABEL IBM_LABEL
      MEAN        = 0.1
      VARIANCE    = 0.01
      TREND       = 0.05
   LABEL MICROSOFT_LABEL [M1]
      MEAN        = 2
      VARIANCE    = 0.02
      TREND       = 0.03
      PAST VALUES = MIC_PAST
   LABEL NOKIA [N_LABEL]
END TREE
```

Labels defined in a tree file (or redefined in a tree section) can be used as array parameters in contract and constraints sections.

3.5 Contract List Section

This section gives the list of contracts involved in a model. The list contains names of contracts separated by commas or new lines. The order of names is not important. Examples 4 and 5 are equivalent.

EXAMPLE 4

```
CONTRACTS
   IBM, MICROSOFT, HITACHI, NOKIA
END CONTRACTS
```

EXAMPLE 5

```
CONTRACTS
   HITACHI
   NOKIA
   MICROSOFT
   IBM
END CONTRACTS
```

3.6 Contract Section

Contract section serves to describing properties of each contract. Now only one contract type is considered (a contract modeling shares on a stock market).

The general construction of contract section is shown below:

```
CONTRACT name
    TYPE SHARE
    PRICING NODE DEPENDENT | PRICING TIME DEPENDENT
    BUY PRICE  = expression  | BUY COMMISSION  = expression
    SELL PRICE = expression  | SELL COMMISSION = expression
    CASH FLOW  = expression
    VALUE      = expression
END CONTRACT
```

The description of fields in a contract section is given in Table 3.7.

With each contract four values are connected: buy price, sell price, cash flow and its value. They may depend solely on time (there is no random factor in this case) or they may be driven by the values "living" on the tree. In the latter case we deal with stochastic uncertainty.

Buy price and sell price may be expressed in two forms: directly by giving proper formula or indirectly by defining buy or sell commission. The two formulations below are equivalent (for buy price, assuming that the contract value is given in the parameter `C1`):

```
BUY PRICE      = 1.01 * C1[N]
BUY COMMISSION = 0.01
```

The mandatory fields are `TYPE` (the only type is now `SHARE`), and `PRICING` (where one of `NODE DEPENDENT` or `TIME DEPENDENT` has to be specified). For others, if they are omitted, zero is taken as a default.

Each contract causes defining two new array variables, the first having the same name as a contract name and denoting the total value of these contracts in the portfolio (for each time stage), and the second having name `NO_OF_`*contract_name* denoting the number of these contracts in the portfolio (for each time stage). These variables may be used only in constraints section (see Example 9).

For example, if there are 10 time periods (numbered from 0 to 9), that for contract `IBM` the following variables are internally defined: `IBM[0:9]` and `NO_OF_IBM[0:9]`.

EXAMPLE 6

name	meaning
TYPE SHARE	type of a contract; share is the only type allowed so far; (mandatory)
PRICING NODE DEPENDENT	a flag denoting that such parameters of a contract as buy price (or buy commission), sell price (or sell commission), cash flow and value are functions of stochastic variables defined by the tree
PRICING TIME DEPENDENT	a flag denoting that such parameters of a contract as buy price (or buy commission), sell price (or sell commission), cash flow and value are deterministically defined for each time stage considered; in this case the use of labels defined in the tree section is illegal
BUY PRICE	defines buy price
BUY COMMISSION	defines buy commission
SELL PRICE	defines sell price
SELL COMMISSION	defines sell commission
CASH FLOW	defines cash flow resulting from a contract
VALUE	defines a contract value used for calculating a portfolio wealth at any time stage

Table 3.7: Fields of contract section

```
CONTRACT IBM
   TYPE SHARE
   PRICING NODE DEPENDENT
   BUY PRICE  = IBM_VALUE[N] * (1 + BUY_COMMISSION)
   SELL PRICE = IBM_VALUE[N] * (1 - SELL_COMMISSION)
   CASH FLOW  = 0
   VALUE      = IBM_VALUE[N]
END CONTRACT
```

EXAMPLE 7

```
CONTRACT IBM
   TYPE SHARE
   PRICING NODE DEPENDENT
   BUY COMMISSION  = BUY_COMMISSION
   SELL COMMISSION = SELL_COMMISSION
   CASH FLOW       = 0
   VALUE           = IBM_VALUE[N]
END CONTRACT
```

EXAMPLE 8

```
CONTRACT COR
   TYPE SHARE
   PRICING NODE DEPENDENT
   BUY COMMISSION  = BUY_COMMISSION
   SELL COMMISSION = SELL_COMMISSION
   CASH FLOW       = 0
   VALUE           = TREE_LABEL_1[N-1] + 2 * TREE_LABEL_2[N-2]
END CONTRACT
```

3.7 Constraints Section

This section is meant to express constraints imposed on a decision problem. The syntax of this section is shown below. Each expression has to be linear with respect to variables. Relation is one of symbols: =, <=, =>.

```
CONSTRAINTS
    FOR N IN {  set  }:  expression relation expression
    FOR ALL N:  expression relation expression
END CONSTRAINTS
```

The example 9 shows how legal constraints can may be written down. Let us assume that the law requires that the amount of shares of each company in our portfolio cannot exceed 5% of the total number of shares issued by a company, and moreover, that the total value of computer companies in the portfolio cannot be greater than 9% of the portfolio value.

EXAMPLE 9

```
CONSTRAINTS
   FOR ALL N: NO_OF_IBM[N]           <= 0.05 * TOTAL_IBM_SHARES
   FOR ALL N: NO_OF_MICROSOFT[N]     <= 0.05 * TOTAL_MICROSOFT_SHARES
   FOR ALL N: IBM[N] + MICROSOFT[N] <= 0.09 * WEALTH[N]
END CONSTRAINTS
```

The next example shows other possibilities of imposing constraints.

EXAMPLE 10

```
CONSTRAINTS
   FOR ALL N: NO_OF_COR[N-1] <= NO_OF_COR[N]
   FOR N IN {7,8,9}: CASH[N] => MANDATORY_RESERVE[N]
END CONSTRAINTS
```

3.8 Initial Values Section

This section provides a way for setting initial cash amount and the initial portfolio. The syntax of this section as well as an example are given below.

```
INITIAL VALUES
    INITIAL CASH =  expression
    NO_OF_contract_name  =  expression
    NO_OF_contract_name  =  expression
    NO_OF_contract_name  =  expression
END INITIAL VALUES
```

EXAMPLE 11

```
INITIAL VALUES
   INITIAL CASH    = 5.4E6
   NO_OF_IBM       = 45
   NO_OF_MICROSOFT = 54
   NO_OF_COR       = 110
END INITIAL VALUES
```

3.9 External Flows Section

The section devoted to modeling external flows is described here. It allows to define external inflows and external outflows separately, just for the sake of readability, although only one of them is sufficient. The syntax of this section is given below.

```
EXTERNAL FLOWS
    EXTERNAL INFLOW  =  expression
    EXTERNAL OUTFLOW =  expression
END EXTERNAL FLOWS
```

The two following examples (12 and 13) are equivalent.

EXAMPLE 12

```
EXTERNAL FLOWS
   EXTERNAL INFLOW  = FORECAST_1[N]
   EXTERNAL OUTFLOW = FORECAST_2[N]
END EXTERNAL FLOWS
```

EXAMPLE 13

```
EXTERNAL FLOWS
   EXTERNAL INFLOW  = FORECAST_1[N] - FORECAST_2[N]
END EXTERNAL FLOWS
```

Yet another example that shows possibilities of defining external flows.

EXAMPLE 14

```
EXTERNAL FLOWS
   EXTERNAL INFLOW  = 2*A1[N-1] - 3*A2[3*N]
   EXTERNAL OUTFLOW = C0 + C * N
END EXTERNAL FLOWS
```

3.10 Goal Section

The purpose of the goal section is to specify goals of decision maker. So far only one formulation of a goal function is supported, the expected value with the risk measure expressed in the form of mean absolute deviation (2.6). The general construction of the goal section is shown below.

```
GOAL
    TYPE MEAN ABSOLUTE DEVIATION
    RISK AVERSION = expression
END GOAL
```

EXAMPLE 15

```
GOAL
   TYPE MEAN ABSOLUTE DEVIATION
   RISK AVERSION = 0.4
END GOAL
```

3.11 Example model description file

```
PROBLEM problem1

TYPE PORTFOLIO OPTIMIZATION

PARAMETERS
   NO_OF_PERIODS    = 5
   IBM_PAST[-3:-1]  = [10, 10, 10]
   COMMISSION       = 0.01 * 2
   MY_RISK_AVERSION = 0.5
   CASH0            = 19E4
   SHARE            = 0.4
   INFLOW[0:10]      = [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
   OUTFLOW[0:10]     = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]
END PARAMETERS

TREE
   FILE NAME data2
   LABEL NOKIA_VALUE [E00001]
      MEAN = 15
      VARIANCE = 1
      TREND = 0.5
      PAST VALUES = IBM_PAST
   LABEL HP_VALUE    [E00002]
   LABEL IBM_VALUE   [E00003]
      MEAN        = 10
      VARIANCE    = 0.1
      TREND       = 1
      PAST VALUES = IBM_PAST
END TREE

CONTRACTS
NOKIA, HP, IBM
END CONTRACTS

CONTRACT NOKIA
   TYPE SHARE
   PRICING NODE DEPENDENT
   VALUE = NOKIA_VALUE[N]
   CASH FLOW = 0.02 * NOKIA_VALUE[N]
END CONTRACT

CONTRACT HP
   TYPE SHARE
   PRICING NODE DEPENDENT
   VALUE = HP_VALUE[N]
   CASH FLOW = 0.02 * HP_VALUE[N]
```

```
END CONTRACT

CONTRACT IBM
   TYPE SHARE
   PRICING NODE DEPENDENT
   VALUE = IBM_VALUE[N]
   CASH FLOW = 0.02 * IBM_VALUE[N]
   BUY COMMISSION = COMMISSION
END CONTRACT

INITIAL VALUES
  INITIAL CASH = CASH0
  NO_OF_NOKIA = 100
  NO_OF_IBM   =  10
  NO_OF_HP    =   5
END INITIAL VALUES

EXTERNAL FLOWS
   EXTERNAL INFLOW  = INFLOW[N] * 1E3
   EXTERNAL OUTFLOW = OUTFLOW[N] * 1E3
END EXTERNAL FLOWS

GOAL
   TYPE MEAN ABSOLUTE DEVIATION
   RISK AVERSION = MY_RISK_AVERSION
END GOAL

CONSTRAINTS
   FOR ALL N: NOKIA[N] <= SHARE * WEALTH[N]
   FOR ALL N: HP[N] <= SHARE * WEALTH[N]
   FOR ALL N: IBM[N] <= SHARE * WEALTH[N]
   FOR ALL N: CASH[N] => 0.05 * WEALTH[N]
END CONSTRAINTS

END PROBLEM
```

Chapter 4

Tree Description File

Tree description file consists of three kinds of sections: `TREE` section, `LABEL` sections and `NODE` sections. The general structure of this file is as follows:

```
TREE:            name
   # PERIODS:    number
   # NODES:      number
   FIRST NODES:  number,number,...,number
   DIMENSION:    number
   LABELS:       name,name,...,name

LABEL:           label
   MEAN:         number
   VARIANCE:     number
   TREND:        number

NODE:            number
   NODELABEL:    label
   VALUES:       number,number,...,number
   PROBABILITY:  number
   PREDECESSOR:  number
   SUCCESSORS:   number,number,...,number
```

The `TREE` section has to begin the file. It describes some general properties of the tree. All is components are mandatory. The field `# PERIODS` tells the last time index covered by the tree (so there is one time period more, namely time period with index 0). The field `# NODES` gives the information how many `NODE` sections are in the file.

The field `FIRST NODES` gives a list of the lowest node numbers for each time index (excluding time index 0). Nodes have to be numbered starting from number 1 for root node (time index 0), ascending, nodes with greater time index have to be assigned greater numbers than nodes with lower time index. The example of proper node numbering is given in Fig. 2.1.

The field `DIMENSION` says the size of data vector assigned to each node. Each dimension of this data vector has its own label declared in the field `LABELS`.

Sections `LABEL` are not mandatory. If given, they deliver mean value, variance, and trend of data subspace described by a given label.

Sections `NODE` describe a node of the tree. Node number has to be given according to rules mentioned above. Node label could be any string. The field `VALUES` gives vector of values assigned to the node. Probability of attaining the node is declared in the field `PROBABILITY`. Two fields: `PREDECESSOR` and `SUCCESSORS` are used to describe the tree structure and give predecessor node number and successor nodes list, respectively. Predecessor of root is 0, and successors of leaves are are denoted by 0, too.

As an example a file describing the tree shown in Fig. 2.1 is given below. Each node has a vector of three elements associated to it.

```
TREE:               example
  # PERIODS:        2
  # NODES:          9
  FIRST NODES:      2     4
  DIMENSION:        3
  LABELS:        Contract_1  Contract_2  Contract_3
LABEL:           Contract_1
  MEAN:          3.00000000
  VARIANCE:      1.00000000
  TREND:        0.000000000
LABEL:           Contract_2
  MEAN:          3.00000000
  VARIANCE:      1.00000000
  TREND:         1.00000000
LABEL:           Contract_3
  MEAN:          5.00000000
  VARIANCE:      1.00000000
  TREND:        -1.00000000
NODE:               1
  NODELABEL:      L00000001
  VALUES:        3.00000000       3.00000000       5.00000000
  PROBABILITY:   1.00000000
  PREDECESSOR:      0
  SUCCESSORS:       2     3
NODE:               2
  NODELABEL:      L00000002
  VALUES:        2.00000000       3.00000000       3.00000000
  PROBABILITY:  0.500000000
  PREDECESSOR:      1
  SUCCESSORS:       4     5     6
NODE:               3
  NODELABEL:      L00000003
  VALUES:        4.00000000       5.00000000       5.00000000
  PROBABILITY:  0.500000000
  PREDECESSOR:      1
  SUCCESSORS:       7     8     9
NODE:               4
  NODELABEL:      L00000004
  VALUES:        1.00000000       3.00000000       1.00000000
  PROBABILITY:  0.166666666
  PREDECESSOR:      2
  SUCCESSORS:       0
```

```
NODE:              5
  NODELABEL:      L00000005
  VALUES:        2.00000000      3.00000000      2.00000000
  PROBABILITY:  0.166666666
  PREDECESSOR:      2
  SUCCESSORS:       0
NODE:              6
  NODELABEL:      L00000006
  VALUES:        3.00000000      4.00000000      3.00000000
  PROBABILITY:  0.166666666
  PREDECESSOR:      2
  SUCCESSORS:       0
NODE:              7
  NODELABEL:      L00000007
  VALUES:        3.00000000      5.00000000      3.00000000
  PROBABILITY:  0.166666666
  PREDECESSOR:      3
  SUCCESSORS:       0
NODE:              8
  NODELABEL:      L00000008
  VALUES:        4.00000000      6.00000000      4.00000000
  PROBABILITY:  0.166666666
  PREDECESSOR:      3
  SUCCESSORS:       0
NODE:              9
  NODELABEL:      L00000009
  VALUES:        5.00000000      7.00000000      5.00000000
  PROBABILITY:  0.166666666
  PREDECESSOR:      3
  SUCCESSORS:       0
```

Chapter 5

Output file

5.1 MPS file format description

5.1.1 Row names

Each row name consists of eight characters. First two identify a constraint and the remaining six are zero or identify a node for which this constraint was generated. Details are described in Table 5.1.

row name		description
`G0`	`000000`	goal function (2.7) coefficients
`G1`	`000000`	equation (2.8)
`G2`	n	equation (2.9) for node n
`01`	n	equation (2.1)
`02`	n	equation (2.3)
`03`	n	equation (2.4)
`04`	n	equation (2.2) for contract 1
$\vdots$	$\vdots$	equation (2.2) for contract j
$J+3$	n	equation (2.2) for contract J
$k+J+3$	n	constraints defined in constraints section

Table 5.1: Row names

5.1.2 Column names

Each column name consists of eight digits. First two identify a variable and the remaining six are zero or denote a node in which this variable is used. Variable identifiers are described in comments in MPS file generated.

Chapter 6

Program invocation

The name of the tool program is `modtool`. It takes the following parameters:

-f *file name* takes file name and produces an MPS file

-h displays help

Bibliography

[BDG$^+$87] Birge, J. R., Dempster, M. A. H., Gassmann, H. I., Gunn, E. A., King, A. J., and Wallace, S. W. A standard input format for multiperiod stochastic linear programs. *Mathematical Programming Society Committee on Algorithms. Newsletter*, 17:1–20, 1987.

[KW94] Kall, P. and Wallace, S. W. *Stochastic Programming.* John Wiley, 1994.