ASB/ACORS 1999 Conference
Halifax, Nova Scotia

H.I. Gassmann
School of Business Administration
Dalhousie University

VIEWS AND REPRESENTATIONS OF STOCHASTIC PROGRAMS[1]

Stochastic programs are difficult objects to work with. This paper describes various views and representations that may make it easier for the modeller to interact with a model, by way of formulating, archiving and visualizing models and model components.

1. Introduction

Stochastic programming problems are mathematical programs in which some of the data are uncertain. A broad classification distinguishes between situations when probability distributions are known (or can be estimated) for all data items, giving rise to optimization under risk, and situations where such distribution information is not known, also called optimization under uncertainty.

One further distinguishes chance-constrained problems, in which constraints involving one or more random variables are only required to hold with a certain (prespecified) probability, and recourse problems, in which corrective action is allowed *after* a particular realization of the random variables has been observed. The recourse decisions add a cost to the objective that can be evaluated only after the random variables have been observed and is normally handled as an *expected* cost.

In some problems the alternating sequence of observing random variables and making decisions based on the actual realizations observed to date is repeated several times, giving rise to multistage stochastic recourse problems. The stages may correspond to explicit time periods, but it is permissible to aggregate several time periods into a single stage, as long as decisions taken at any time depend only on the information available beforehand. This is called the principle of *nonanticipativity*. In the case of discrete distributions *event trees* give a convenient way to picture the flow of information (i.e., what the decision maker knows when). Event trees differ from the widely used decision trees in that they record only the random events; the decisions taken are implicit. A small event tree is depicted in Figure 1.

This paper is concerned with linear multistage stochastic programs, in which all the constraints and the objective are linear (affine) functions of the decision variables. Working with

[1] This research was supported in part by a grant from the National Sciences and Engineering Research Council of Canada (NSERC)

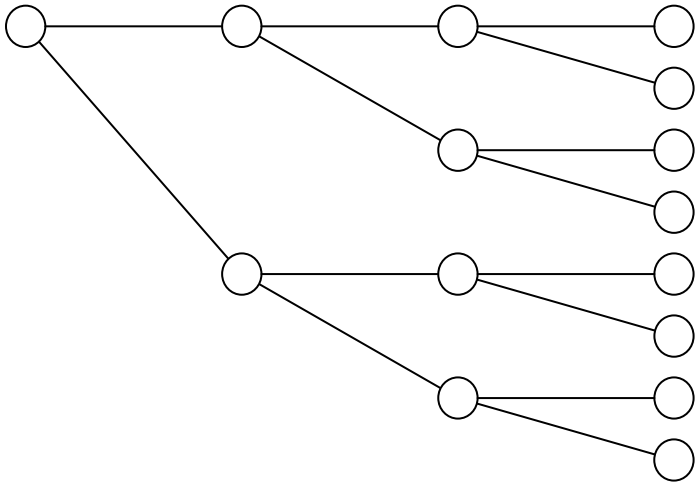

Figure 1. A small event tree

such problems can be highly nontrivial; not just their solution but even such mundane tasks as formulation, storage and data handling can be difficult and time consuming, mostly because of the large problem size.

The remainder of this paper describes some ways to facilitate the mentioned model management functions. Section 2 describes an input format based on the MPS format for linear programs that can be used for problem formulation. Section 3 introduces some components and functions of an algebraic modelling language for the same purpose, and Section 4 features different database tools for visualizing both data and solutions.

We begin by formulating two stochastic linear recourse problems, which will be used for illustrative purposes. The first is a stochastic version of a transportation problem, given mathematically as

$$\min \quad \sum_{i,j} c_{ij} x_{ij} \quad + \quad \sum_{w} p_w \left[\sum_{j} \left(q_j^+ y_{jw}^+ + q_j^- y_{jw}^- \right) \right] \tag{1.1}$$

$$st \quad \sum_{j} x_{ij} \qquad\qquad \leq \quad s_i \quad \text{for all } i \tag{1.2}$$

$$\sum_{i} x_{ij} \qquad -y_{jw}^+ + y_{jw}^- \qquad = \quad d_{j,w} \quad \text{for all } j,w \tag{1.3}$$

$$x_{ij} \quad \geq \quad 0 \quad \text{for all } i,j \tag{1.4}$$

$$y_{jw}^+, y_{jw}^- \quad \geq \quad 0 \quad \text{for all } j,w. \tag{1.5}$$

Here d_{jw} is the random demand realized in demand centre j under scenario w, p_w is the probability of observing outcome or scenario w, and q_j^+ and q_j^- are the costs incurred in dealing with over- and underallocations if the total shipment into demand centre j exceeds or falls short of the realized demand.

It should be remarked that this mathematical formulation defines sets and indices (sources, destinations, scenarios), variable classes (x, y^+, y^-) and general relationships (such as the generic product balance equations (1.3)). No mention is made of the actual set memberships (i.e., what the sources, destinations and scenarios actually are) or the values of the coefficients; that is left for later. This is an example of the separation of model and data, which we will return to in Section 3.

The second problem is a multistage financial planning problem. The full formulation is given in Gassmann and Ireland (1995); we only summarize the most salient features here. At each stage the decision-maker selects borrowing and repayment decisions in a number of different debt instruments issued in different markets, subject to cash requirement, market saturation and other institutional and regulatory constraints. Uncertainty enters in the form of interest and exchange rate risk. The objective is to minimize the expected value of the debt outstanding at the end of the planning horizon.

In block schematic form, this problem can be written as

$$\min \quad \sum_{s \in N(T)} p_s x_{Ts} \tag{2.1}$$

$$st \quad A_0 x_0 = b_0 \tag{2.2}$$

$$B_{ts} x_{t-1,a(s)} + A_{ts} x_{ts} = b_{ts} \quad \text{for } s \text{ in } N(t), t=1,\ldots,T \tag{2.3}$$

$$x_0 \geq 0 \tag{2.4}$$

$$x_{ts} \geq 0 \quad \text{for } s \text{ in } N(t), t=1,\ldots,T, \tag{2.5}$$

where it is assumed that the nodes in the event tree are numbered in some fashion (for instance, stage by stage), such that the root node is numbered 0 and for each node $s > 0$, $a(s)$ gives the ancestor node in the previous stage. $N(t)$ denotes the collection of all nodes in stage t.

2. The SMPS format

It is very important to have a commonly accepted way to describe a stochastic program. Archiving, algorithmic development and benchmarking, and test problem collection all depend on it. Linear programming has used the MPS format (see, e.g.,IBM 1972a,b,1996) as a de facto standard for about forty years. In this section we describe a modification of the MPS format designed to handle stochastic extensions. This so-called SMPS format is described in full detail in Gassmann and Schweitzer (1996), and is based on earlier work by Edwards *et al*. (1985) and Birge *et al*. (1988).

The key idea is that multistage stochastic problems contain special structure of two types, temporal information to describe the time stage each row and column of the problem belongs to, and stochastic information to describe the distributions of the random coefficients. Because the random information is revealed gradually over time, the stochastic structure is a secondary structure, which cannot be conveyed accurately without describing the temporal structure first. All this information is collected in the SMPS format in three separate files, called the core file, the time file, and stochastic file (or stoch file, for short).

The core file is in MPS format and fixes the problem dimensions, row and column names, deterministic coefficients, and representative values for all the random elements. The latter could be the means, but that is not a strict requirement. (In this simplified exposition we assume that the dimensions of the problem are deterministic; the SMPS format has special devices for dealing with stochastic problem dimensions, such as trap states or variables that are only activated if a particular set of realizations has been observed.)

One important simplifying assumption is that the rows and columns have been sorted into temporal order, so that all first-stage rows are mentioned in the core file before the appearance of the first row of the second stage, and so on. (And similarly for the columns.) This allows for a very simple description of the time structure, by simply marking the first row and column for each stage. This information is contained in the time file.

The random structure is described in the stoch file. The goal is to arrange the possible realizations into an event tree, which can be set up implicitly or explicitly. Explicit event trees are built one scenario at a time, by specifying branching probabilities along with all the information that is particular to each scenario. Explicit event trees are finite by definition. Implicit event trees are generated by the system; the user merely supplies information about marginal distributions governing one or more random elements. Implicit trees can at present only be dealt with if the random elements are independent from one stage to the next, but the format provides support for continuous distributions that are discretized (or sampled) only during the solution phase.

Figure 2 shows a sample stoch file for the transportation problem, in which the demands at four destinations are random, independently of each other. (Depending on the information given in the core file, these may be the only destinations, or there may be other destinations at which the demand is deterministic.) Lines 2 to 5 indicate that the demand at destination 'TRD' is discrete and takes on values of 150, 180 and 120 units with probability 0.25, 0.5, 0.25, respectively. Similarly the demand at destination 'KRS' is assumed uniformly distributed on the interval [80,110], the demand at 'STV' is normally distributed with mean 100 and variance 100, and the demand at 'LIL' follows a distribution described by the (user-supplied) subroutine `My_Jiffy` (which must be distributed with the data files to completely specify the problem instance).

```
NAME    Transportation problem
INDEP         DISCRETE
    RHS       DEMTRD      150.0         0.5
    RHS       DEMTRD      180.0         0.25
    RHS       DEMTRD      120.0         0.25
INDEP         UNIFORM
    RHS       DEMKRS       80.0         110.0
INDEP         NORMAL
    RHS       DEMSTV      100.0         100.0
INDEP         My_Jiffy
    RHS       DEMLIL       80.0         100.0
ENDATA
```

Figure 2. Stochastics file for the transportation problem

3. Algebraic modelling languages

Algebraic modelling languages have been used for some years to assist with the generation of linear (and nonlinear) mathematical models. Their aim is to mimic as closely as possible the notation used by the modeller to describe the model in conventional mathematical notation. Algebraic modelling languages often feature the separation of model and data, as mentioned in the introduction. Commonly used algebraic modelling languages are AMPL (Fourer et al. 1993), GAMS (Brooke et al. 1988), MPL (Kristjansson 1993), MODLER (Greenberg 1993), and AIMMS (Bisschop and Entriken 1993).

In as much as algebraic modelling languages can be used to formulate large scale linear programs, they can be used to set up stochastic programs with finite discrete distributions (in explicit scenario form). For example, the transportation problem (1) can be formulated in AMPL as shown in Figure 3. A data section to define a particular instance is displayed in Figure 4.

```
set ORIG;   # origins
set DEST;   # destinations
set SCEN;   # scenarios

param supply {ORIG}      >= 0;   # amounts available at origins
param demand {DEST,SCEN} >= 0;   # amounts required at destinations
param prob   {SCEN}      >= 0;   # scenario probabilities
param cost   {ORIG,DEST} >= 0;   # shipment costs per unit
param shortcost {DEST}   >= 0;   # shortage costs at destinations
param surplcost {DEST}       ;   # surplus  costs at destinations

var   Trans {ORIG,DEST}  >= 0;   # units to be shipped
var   Short {DEST,SCEN}  >= 0;   # units short in each scenario
var   Surpl {DEST,SCEN}  >= 0;   # units long in each scenario

minimize total_cost:
   sum {s in ORIG, d in DEST} cost[s,d] * Trans[s,d] +
   sum {d in DEST, w in SCEN} prob[w]*(shortcost[d]*Short[d,w] +
                                       surplcost[d]*Surpl[d,w]);

subject to Supply {s in ORIG}:   sum {d in DEST} Trans[s,d] <= supply[s];

subject to Demand {d in DEST, w in SCEN}:
   sum {s in ORIG} Trans[s,d] + Short[d,w] - Surpl[d,w] = demand[d,w];
```

Figure 3. An AMPL formulation of the transportation model

```
param: ORIG:  supply :=         # defines set "ORIG" and param "supply"
        Oslo      250
        Bergen    100;

param: SCEN:  prob :=           # defines set "SCEN" and param "prob"
        High      0.25
        Medium    0.5
        Low       0.25;

param: DEST:  shortcost   surplcost  :=    # defines set "DEST" and
                                           # parameters "surpl" and "short"
     Stavanger     15       5
     Trondheim     10       0
     Lillehammer   20      -5;

param cost: Stavanger   Trondheim   Lillehammer :=
   Oslo        10          15          10
   Bergen      10          10          20;

param demand:  High      Medium    Low  :=
   Stavanger   110       100        90
   Trondheim   165       150       135
   Lillehammer  88        80        72;
```

Figure 4. A data file to accompany the AMPL model of Figure 3

Since the financial planning model is a multistage model, it is by necessity more involved than the two-stage transportation model. The full formulation of the financial planning model has been omitted for space reasons, but some key points should be noted. First, each scenario is given

a start time as well as a parent scenario from which it branches. (The root scenario has no parent.) To make the correspondence to the event tree of Figure 1 completely explicit we use the convention that slanted arcs indicate branching, while horizontal arcs connect two stages on the same scenario. The root scenario tus corresponds to the topmost scenario and the second scenario branches from it starting in stage 4.

An inventory balance equation might then be written in AMPL in the form suggested in Figure 5.

```
balance { (j,t) in event_nodes} :

cash[j,t] =                      ; cash on hand in period t
   sum {k in debt_types}
      exch_rate[k,j,t] * ( (1-issue_cost[k,t])*borrow[k,j,t]
                                              ; new borrowing

        - sum { s in 0..t-1} (int_rate[k,j,s,t] *
            (if t = 1                         ; interest payments
             then init_debt[k]                ; on outstanding debt
             else outst[k,previous[j,t],s,t-1])

        + ret_cost[k,j,s,t]*retire[k,j,s,t] ))    ; repayments

   + ( if t = 1                      ; cash from previous period
       then init_cash
       else (1 + cash_int[j,t]) * cash[previous[j,t],t-1]);
```

Figure 5. AMPL formulation of a constraint involving random variables

Here `event_nodes` is a subset of the cartesian product (`periods` × `scenarios`), corresponding exactly to the nodes in the event tree. (This set can be constructed automatically once the parent relation and the starting time are given for each scenario; the details are omitted.) The crux of the formulation is AMPL's ability to use the predefined function `previous[j,t]` in place of an indexing variable. The definition of `previous[j,t]` is simply `j` if scenario *j* contains a node in the event tree in the previous period (i.e., if period *t* is not the starting time of scenario *j*) and is `parent[j]` otherwise. This particular way of thinking about scenarios is convenient for data handling and uses a non-redundant representation of the data. We will call this the compressed form of the event tree, and note that nonanticipativity of decision variables is handled implicitly.

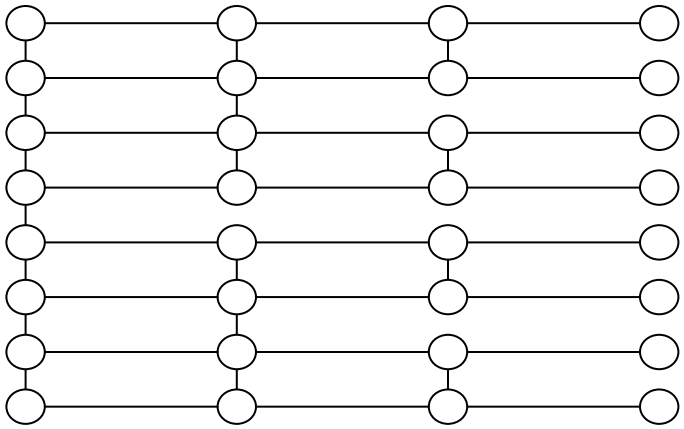

Figure 6. A small event tree in expanded form

But each scenario also represents a path from the root node to one of the leaf nodes. For displaying a time series of one data element or decision variable (such as the holdings in a partic-

ular financial instrument) over time, the expanded form of the event tree is more convenient. In the expanded form, each data element and decision variable is replicated in each scenario, and nonanticipativity is enforced through explicit constraint. Figure 6 shows the expanded view of the event tree given originally in Figure 1. The vertical lines indicate nodes whose decision variables must be made to agree by explicit constraints.

It is even possible to define implicit event trees in AMPL, provided that the scenarios are identified by numbers instead of names. The reason for this is that the branching information can be computed using modular arithmetic.

Figure 7 shows again the small event tree; this time the scenarios have been numbered consecutively from 1 to 8. We assume that there is one random variable in each stage. Each random variable takes two possible values and is independent of the previous stages. Note that the odd-numbered scenarios use the first outcome of the stage 4 random variable and start prior to stage 4, the even-numbered scenarios use the second outcome and start in stage 4. Similarly, if the scenario number $s \equiv 1 (\mathrm{mod} 4)$, then the scenario uses the first outcome of the third-stage random variable, and it started prior to stage 3, etc. A full example of how to implement this idea in AMPL is contained in Gassmann and Ireland (1995).

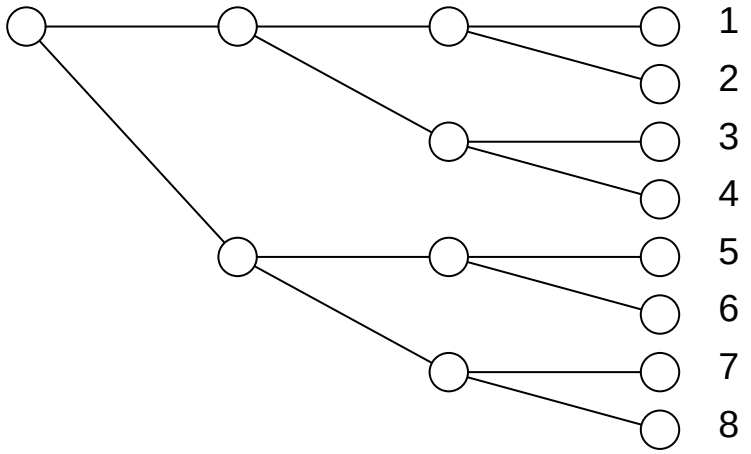

Figure 7. Event tree with numbered scenarios

4. Database views

Some algebraic modelling languages (such as AMPL Plus (Compass Modeling Solutions 1995) and AIMMS (Bisschop and Entriken 1993)) allow data to be contained in a database. Through suitable queries, parameter and coefficient values are read form the database, and solutions returned from the solver can be written back into the database. A recent paper by Fourer (1997) explores the use of databases in mathematical modelling and identifies several important advantages. First, databases are well accepted and understood by managers, making mathematical models in general and stochastic programs in particular accessible to a wider audience.

Second, databases make it easy to eliminate data redundancies and thus reduce the possibility of error. And finally, report generation is made much easier through the use of built-in reporting and graphing capabilities. In other words, one would store the event tree in compressed form. Queries can then be used to switch to expanded form whenever needed.

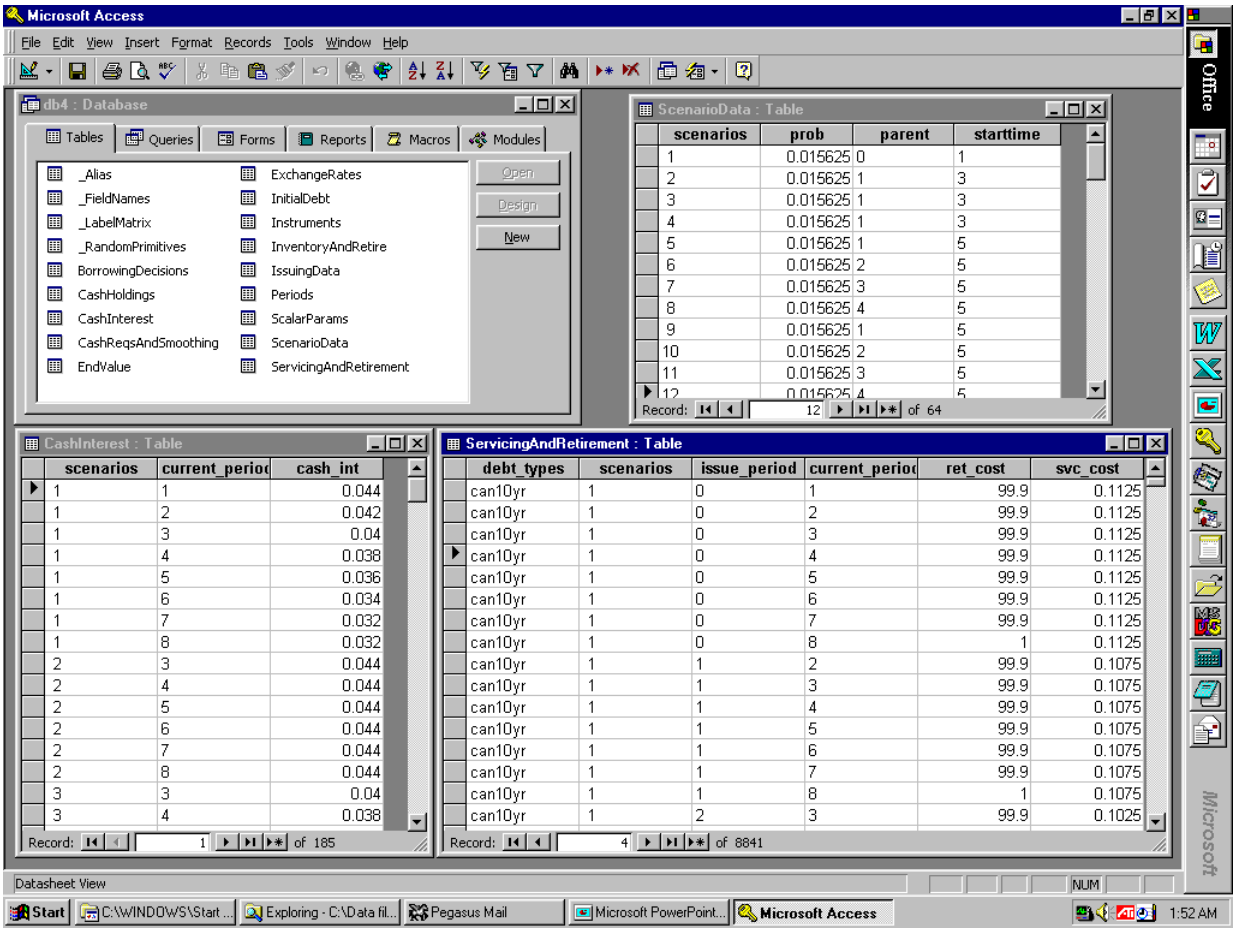

Figure 8. Database structure for a financial planning model

The approach is illustrated in the two screen captures of figures 8 and 9. The first figure shows the database structure for the financial planning model, containing tables for data and solution variables as well as four auxiliary tables (whose names start with the underscore character). Two data tables are also shown to illustrate how the data are stored in compressed form. (There is no entry for *cash_int* in the *CashInterest* table for scenario 2 in *current_period* 1, since scenario 2 only starts in period 3.)

The second screen shows how data for a particular scenario (selected by the user) can be retrieved from the database and displayed in a line graph. (Other graphical representations are just as easy, limited only by the capabilities of the database.) This figure also shows how the data values in the different periods must be retrieved (by a SQL query) from several scenarios. The event tree for this particular problem is depicted (in part) in Figure 10.

5. Conclusions

This paper showed a number of ways to represent stochastic programming problems and their components. These views may aid the modeller in setting up stochastic models; they may also help the user visualize solution vectors and data items. Tools such as these may make stochastic problems more accessible and easier to work with.

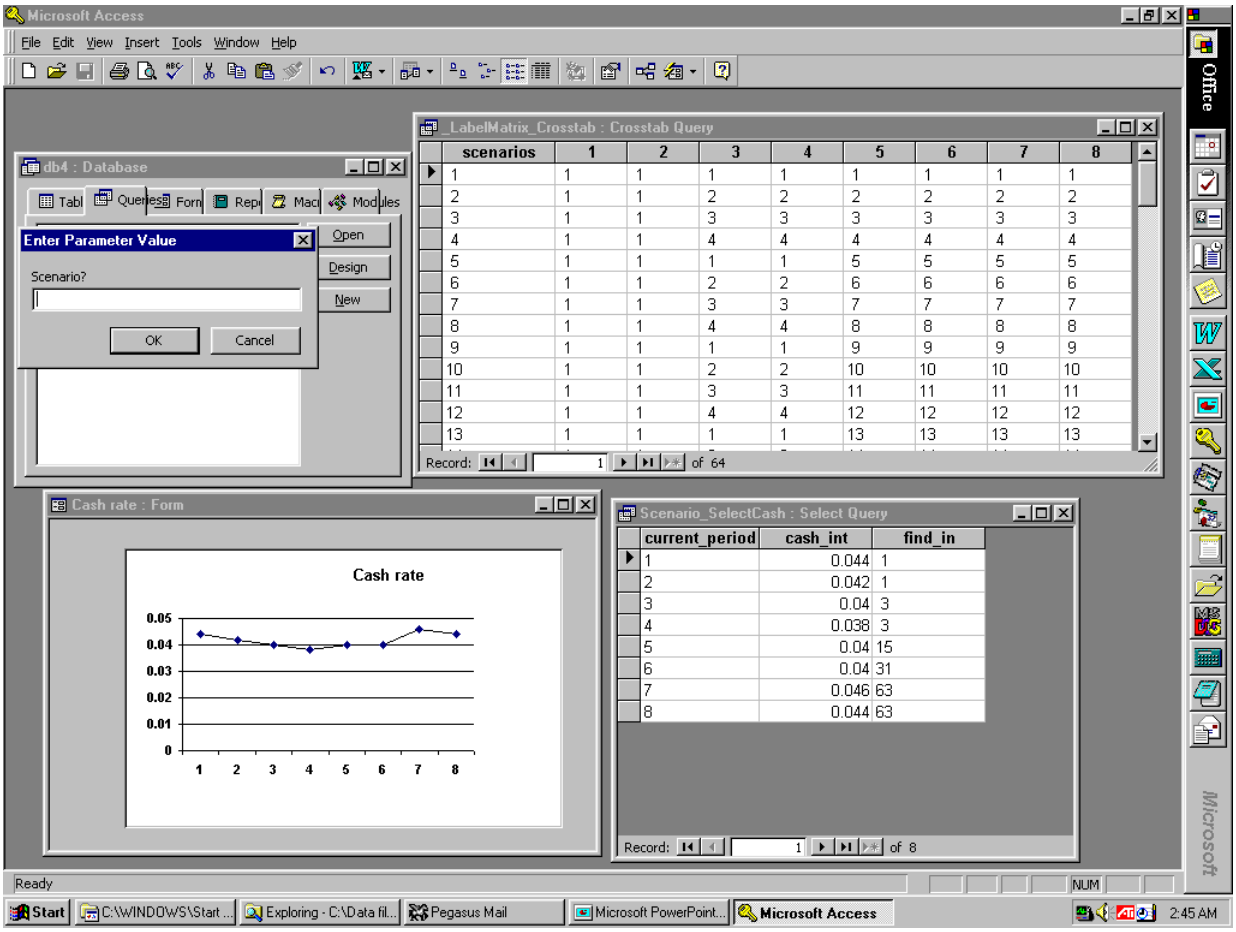

Figure 9. Graphical representation of a data item

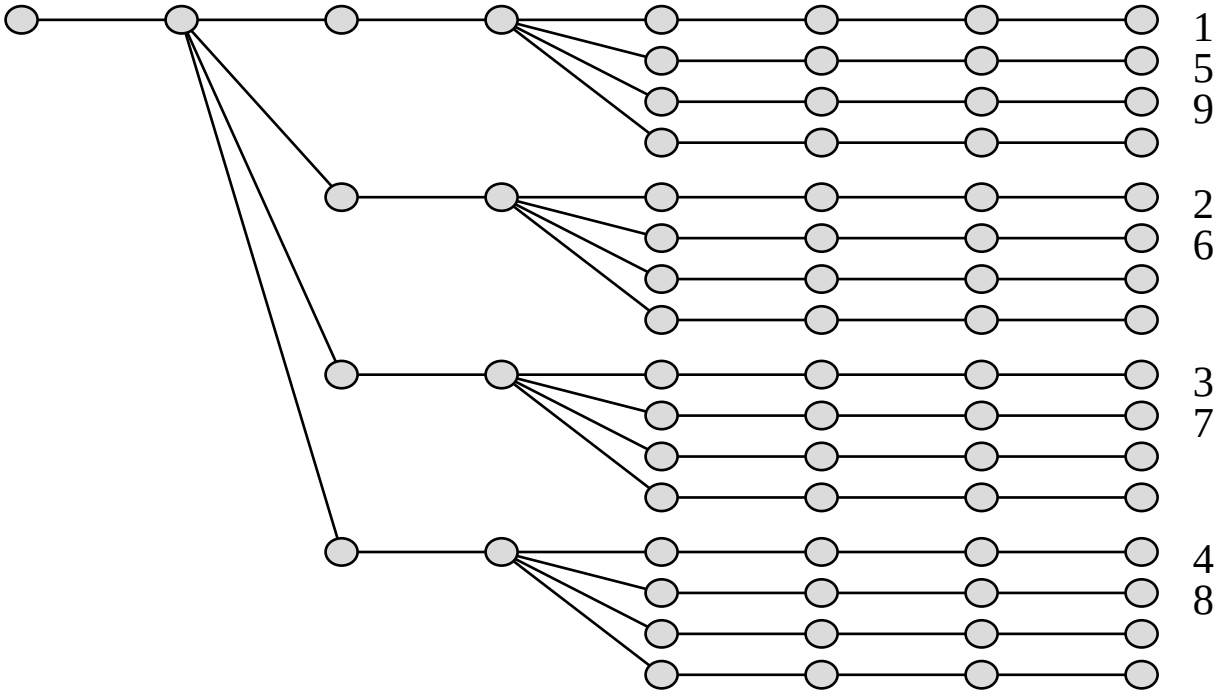

Figure 10. A partial event tree for the financial planning problem

References

Birge, J.R., Dempster, M.A.H., Gassmann, H.I., Gunn, E.A., King, A.J. and Wallace, S.W. (1988) "A standard input format for multiperiod stochastic linear programs". *Committee on Algorithms Newsletter* **17**, 1–19.

Bisschop, J. and Entriken, R. (1993) *AIMMS: The Modelling System*. Paragon Decision Technology, Haarlem, The Netherlands.

Brooke, A., Kendrick, D. and Meeraus, A. (1988) *GAMS -- A User's Guide*. The Scientific Press, Redwood City, California.

Compass Modeling Solutions (1995) *Using AMPL Plus*. Reno, Nevada.

Edwards, J., Birge, J., King, A. and Nazareth, L. (1985) "A standard input format for computer codes which solve stochastic programs with recourse and a library of utilities to simplify its use". Working Paper WP-85-03, International Institute for Applied Systems Analysis, Laxenburg, Austria.

Fourer, R. (1997) "Database structures for mathematical programming models". Decision Support Systems 20, 317–344.

Fourer, R., Gay, D.M., and Kernighan, B.W. (1993) AMPL: *A Modelling Language for Mathematical Programming*. The Scientific Press, San Fransisco, California.

Gassmann, H.I. and Ireland, A.M. (1995) "Scenario formulation in an algebraic modelling language". *Annals of Operations Research* **59**, 45–75.

Gassmann, H.I. and Ireland, A.M. (1996) "On the formulation of stochastic linear programs using algebraic modelling languages". *Annals of Operations Research* **64**, 83–112.

Gassmann, H.I. and Schweitzer, E. (1996) "Proposed extensions to the SMPS input format for stochastic programs". Working Paper WP-96-1, School of Business Administration, Dalhousie University, Halifax, Nova Scotia.

Greenberg, H.J. (1993) *Modeling by Object-Driven Linear Elemental Relations: A User's Guide for MODLER*. Kluwer Academic Publishers, Dordrecht.

International Business Machines. (1972a) "Mathematical Programming Subsystem — Extended (MPSX) and Generalized Upper Bounding (GUB) Program Description", Document number SH20-0968-1.

International Business Machines. (1972b) "Mathematical Programming Subsystem — Extended (MPSX) and Mixed Integer Programming (MIP) Program Description", Document number GH19-1091-0.

International Business Machines. (1996) "Passing your model to OSL using Mathematical Programming System (MPS) format. [web page] http://www.research.ibm.com/osl/ekkgc10.html. [Accessed 28 April 1999].

Kristjansson, B. (1993) *MPL Modelling System User Manual (Version 2.8)*. Maximal Software, Inc., Arlington, Virginia.