Annals of Operations Research 59(1995)45–75

Scenario formulation in an algebraic modelling language

H.I. Gassmann and A.M. Ireland

School of Business Administration, Dalhousie University, Halifax, Nova Scotia, Canada B3H 1Z5

Algebraic modelling languages have simplified management of many types of large linear programs but have not specifically supported stochastic modelling. This paper considers modelling language support for multistage stochastic linear recourse problems with finite distributions. We describe basic language requirements for formulation of finite event trees in algebraic modelling languages and show representative problems in AMPL using three commonly used scenario types.

1. Introduction

The large mathematical programs used to solve real-world optimization problems often require daunting efforts in initial model formulation and matrix generation. Algebraic modelling languages have simplified these processes by providing English-like model specifications and easily modifiable data structures, making the construction of large models easier under the time and cost pressures that arise in business settings.

Stochastic programming models extend the scope of linear and nonlinear programming to include probabilistic or statistical information about one or more uncertain problem parameters. Chance-constrained programming accomplishes this by setting reliability levels within which each constraint or group of constraints must be satisfied. Stochastic programming models with recourse define multiple scenarios and generate contingent solutions depending on actual observations of random events in the future. The increased complexity of both types of models compared to deterministic models poses an even greater barrier to their widespread use, yet research in stochastic programming and algebraic modelling languages has not so far addressed the formulation of stochastic models in these languages.

This paper is concerned primarily with scenario-based linear programs with recourse (SLP). It identifies basic requirements for scenario formulation in algebraic modelling languages and describes our formulation of representative problems in one algebraic modelling language, AMPL. We first outline basic functional requirements

for SLP model specification. We then describe three types of scenarios used in SLP models and show that each can be specified in the AMPL language as it now exists, although the formulation has some artificial aspects due to limitations in the language. This is most evident in problem formulations which attempt to capture the stochastic elements implicitly by giving distributions as opposed to stating the full scenario tree explicitly. Since AMPL was designed to formulate and model deterministic problems, these difficulties should come as no surprise and point towards possible extensions to the language. In this context we discuss related problems in scenario formulation and management which must be addressed in comprehensive systems supporting stochastic linear programming.

2. Algebraic modelling languages

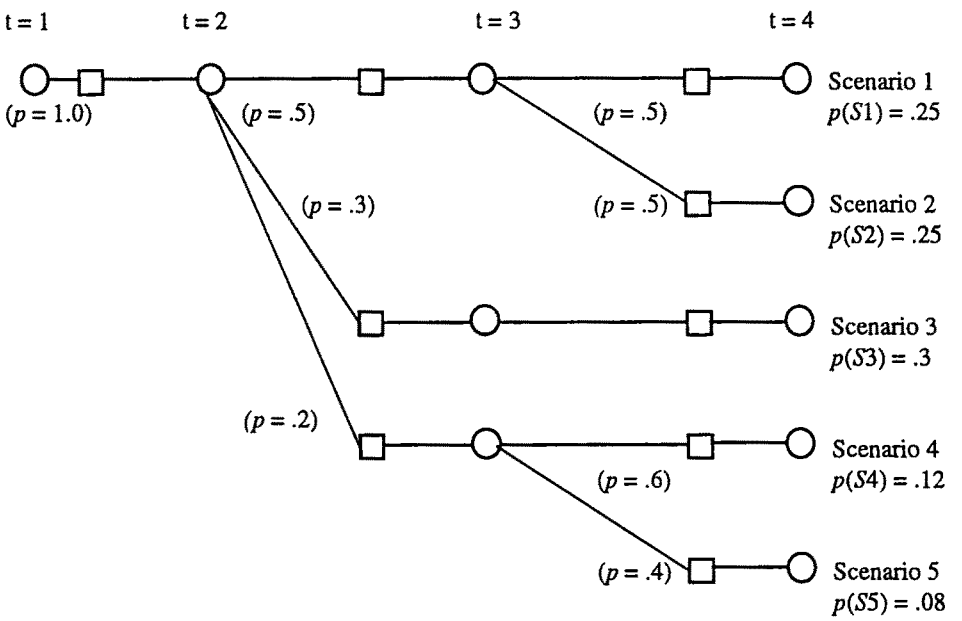
Algebraic modelling languages such as GAMS (Bisschop and Meeraus [2], Brooke et al. [3]) and AMPL (Fourer et al. [5]) are designed to allow modellers to represent linear programming models in statements that closely resemble their mathematical specifications. As described by Greenberg [7], these languages (a) are declarative, specifying *what* is being computed rather than *how* it is done; (b) make extensive use of domains over sets, corresponding to the indexing found in algebraic model specifications, and (c) represent models by rows (constraints) rather than columns (activities). In Fourer's terms, they represent a model in the "modeller's form", subsequently translating it automatically into the "algorithm's" (computer solvable) form, carrying out data consistency checks and automatic matrix generation directly from the language and data specifications.

For example, a model specification in AMPL consists of definitions of index sets, numerical parameters and decision variables, followed by objective and constraint statements taken directly from the model's mathematical representation with changes in syntax that translate the mathematical notation into text. This specification defines a symbolic class of models; data sets to instantiate the model are entered separately as text or database records, and the translation step to MPS format for solver input is handled directly by the AMPL software. Appendix A shows the algebraic formulation, AMPL specification and AMPL data file for a straightforward deterministic production problem taken from Fourer et al. [5].

Algebraic modelling languages have been developed over a number of years to handle many complexities in the formulation of large linear programs such as management of multiple runs, network structure definitions and piecewise linear functions (Fourer [4]). They also provide convenient extensions of the algebraic formulation such as computed parameters, which eliminate the need, for separate calculations to transform raw data into coefficients (Fourer et al. [5]). However, they have not so far explicitly considered specification and management of the structures and data required for multiple scenarios in SLPs – models which could greatly benefit from easier ways to handle their complexity.

3. Scenario formulation requirements

Scenario-based stochastic linear programs optimize under uncertain future conditions by producing contingent decisions over a number of future scenarios. A typical set of scenarios, arranged in a branching, probabilistic tree, is shown in figure 1. Each node in the tree corresponds to a time period with an associated problem state. At each branch point a random event occurs, and the next time period is associated with a number of new states based on the realizations of the random variable with the discrete distribution described by the (conditional) branch probabilities. A *scenario* can be informally defined as a path through the tree from the root to a leaf; the scenario's probability is the probability of all its events occurring.



Notes

○ denotes a rate event.

□ denotes a set of decisions. There are no decisions in the final period

p denotes the conditional probability of a rate event given prior events.

$p(Sn), n = 1,\ldots,5$ denotes the scenario (path) probability.

Figure 1. Branching rate event probability tree.

Within an SLP, parameter values and decision variables are defined for each node in the scenario tree. Because parameters and decisions are contingent on the current problem state which includes the effects of all prior decisions, which in turn

are dependent on all prior events, the current parameters and decisions can be said to be contingent on all prior events.

On the other hand, scenarios share information up to a point where branching occurs. Since the events up to the branching point are insufficient to distinguish between the scenarios, the decisions taken up to that point must also be the same. This concept of *nonanticipativity* is a central theme in stochastic programming. To illustrate these concepts, we consider an investment problem using the tree in figure 1 to describe future interest rate movements. Here earnings parameters would be defined for each time period in each scenario, and contingent investment decisions would be produced for each time period in each scenario. Nonanticipativity requires that all five scenarios share the same parameter and decision values in the first period, that scenario 2 have the same parameters and decisions as scenario 1 in periods 1 and 2, and that scenario 5 be indistinguishable from scenario 4 before period 3.

An SLP therefore differs from a deterministic LP in that parameters, decision variables and constraints have multiple realizations for time periods beyond the first (root) period which correspond to the multiple scenarios active at a given time. Formulating such a model requires (a) defining its scenario structure and (b) specifying the scenario dependencies of its coefficients, variables and constraints as modifications of the deterministic model form.

Typically, scenarios are described in a model's algebraic formulation as sequences of events or realizations of random variables over time. However, scenario formulation is complicated by the fact that in many cases these realizations can be computed from a few parameters rather than explicitly stated as model data, in the same way that coefficients can often be computed from raw data. Our first requirement for scenario formulation is therefore that:

R1: Scenarios should be able to be parametrically specified and computed by the matrix generator, as well as defined explicitly when necessary.

Furthermore, both event sequences and parametric specifications include implicit redundancies when paths overlap, as they do between the root of a scenario tree and branch points. Because formulation and matrix generation will be more efficient if these redundancies are eliminated, our second requirement is that:

R2: Scenario representations should be minimum representations of the problem (the most compact possible) which fully describe scenario structure and scenario-dependent data.

The second requirement essentially means that nonanticipativity is handled *implicitly* by suppressing all decision variables, constraints and parameters that do not correspond to existing nodes in the event tree. For computational efficiency, especially of interior point algorithms, it is sometimes expedient to treat these nonanticipativity constraints *explicitly* and introduce redundant variables (see, for example, Lustig et al. [10]).

For greater understanding of these requirements, we will now explain the formulation of three types of scenarios and specify a model in AMPL for each.

4. Scenario formulation examples

We have initially identified five types of random structures used in SLP models and have attempted to specify three of them in AMPL. We have based all specifications on the event-sequence-based view of scenario trees, used by modellers, rather than the node-and-arc tree representation, normally used computationally and in SLP solvers such as MSLiP (Gassmann [6]). To eliminate redundancy, we view a scenario tree as consisting of the following:

(a) a *base scenario* with problem states defined for all time periods in the planning period, including the root (present) period,

(b) one or more additional scenarios. Each additional scenario shares at least the root period problem state with the base scenario and branches from a *parent* scenario so that it has a distinct new problem state at its *start time*, some time period after the initial period.

For each time period after the start time up to this scenario's horizon – which may differ from the horizon of the base scenario – decision variables and constraints must then be set up, even if the parameters for future periods coincide with those of the base scenario.

A scenario tree structure is therefore defined by specifying the number of time periods, the scenario index set, and parent names, start time values and probabilities for all additional scenarios. (The probabilities can be given as path probabilities or can be conditional on prior events in the parent scenario.) Parameters, variables and constraints are then set up for all time periods in each scenario which are not shared with its parent scenario. (Note that this deals with nonanticipativity implicitly in a very simple and efficient way.)

4.1. TYPE 1: ARBITRARY SCENARIO STRUCTURE WITH FIXED HORIZON

The first type of problem assumes a common fixed time horizon for all scenarios, but otherwise allows for an arbitrary scenario structure which is explicitly stated, along with all required data. Random variable distributions are assumed to be dependent on both the time period and prior history, which determines the current position in the tree. This can happen, for example, in planning problems when assumed scenarios reflect major, unique future events.

An example of this type of problem is found in the SLP model used in the MIDAS decision support system for debt management (Ireland [8]), shown in appendix B. This model is designed to optimize corporate borrowing decisions over a planning

horizon by choosing from a number of debt types and investment types, denoted by index k, issued in various time periods, denoted by index s, to meet cash requirements in each future time period, denoted by index t. This model uses an arbitrary tree structure reflecting projected future interest and exchange rate movements as specified by the model's user, a corporate debt manager. Because these rate movements are influenced by major external events, the number and timing of branches depend on the manager's assumptions about future events and do not follow a predictable, computable pattern.

In appendix B, scenarios are algebraically specified for this type of model as event sequences $e_j = (e_{j1}, e_{j2}, \ldots, e_{jT})$, for $j = 1, \ldots, J$ over a time series $1, \ldots, T$. This notation does not represent the branching tree structure described above, and it does not identify events which are shared across scenarios and should therefore be specified only once in the modelling language. We therefore suggest a modified form of event sequences which leads to a nonredundant AMPL specification.

We follow the ideas put forth in Lane and Hutchinson [9]. Among all scenarios that share data in a particular period t, we choose (arbitrarily) the one with the lowest number and call its event e_{jt} a *representative event*. Representative events are rendered with capital letters; the sets of all representative events are denoted E_t, for $t = 1, \ldots, T$.

For example, in the tree of figure 1, we have $\mathrm{E}_1 = \{E_{11}\}$, $\mathrm{E}_2 = \{E_{12}, E_{32}, E_{42}\}$, $\mathrm{E}_3 = \{E_{13}, E_{23}, E_{33}, E_{43}, E_{53}\}$, $\mathrm{E}_4 = \{E_{14}, E_{24}, E_{34}, E_{44}, E_{54}\}$. For each scenario j the representative event in period T is E_{jT}, but in prior periods it is possible to have other scenario indices appearing. Each such scenario is an ancestor of scenario j. This ancestor relationship induces a partial order on the set of scenarios, and the immediate predecessor of a scenario is referred to as its *parent scenario*. In the tree of figure 1, scenario 1 is parent to scenarios 2, 3 and 4, while scenario 5 has two ancestors, 1 and 4, with scenario 4 being the parent scenario.

We can then represent the tree as the set of event sequences:

$$\begin{aligned}
&\{(E_{11}, E_{12}, E_{13}, E_{14}),\\
&(E_{11}, E_{12}, E_{23}, E_{24}),\\
&(E_{11}, E_{32}, E_{33}, E_{34}),\\
&(E_{11}, E_{42}, E_{43}, E_{44}),\\
&(E_{11}, E_{42}, E_{53}, E_{54})\}.
\end{aligned}$$

The tree structure is implicit since branches are inferred from the data for a specific problem when event realizations are the same for multiple scenarios, as in the overlap of E_{11} in all scenarios and E_{42} in scenarios 4 and 5. The specification is still redundant to the extent that scenario events before branch points are duplicated in multiple scenarios.

Our AMPL specification for the MIDAS model, and a simple instance using a three-period tree with one branch after period 2, is given in appendix B. It is possible to represent this model in AMPL nonredundantly if, as explained above, we

conceptualize the scenario tree as a base scenario with new scenarios branching off below existing ones; this scenario structure is explicitly specified, and careful index restrictions limit parameters, decision variables and constraints to time periods not shared with previously specified periods in parent scenarios.

The following definitions are therefore added to what would have been a deterministic model:

(a) the index set scenarios, used to index scenario-dependent model components;

(b) the parameter prob, indexed over scenarios: the path probability for each scenario;

(c) the parameter parent, indexed over scenarios: the path from which a scenario branches;

(d) the parameter starttime, indexed over scenarios: the first time period in which the scenario has a problem state which differs from that of the parent scenario;

In addition to these user-specified data items, the following are computed by AMPL to support the nonredundant model representation:

(e) The index set event_nodes, defined as the members (j,t) of the cartesian product scenarios × periods satisfying the relationship

$$t \geq \text{starttime}[j];$$

(f) the index set evolutions, used for triple indexing on variables such as outst and retire which depend not only on the node when the retirement decision is made, but also on the period when the debt was first issued, defined as follows:

$$\text{evolutions} := \{(j,s,t) \in \text{scenarios} \times \text{periods} \times \text{periods}: s \leq t \text{ and } t \geq \text{starttime}[j]\};$$

(g) the parameter previous, indexed over event_nodes to describe the way information is inherited from the previous period, defined by

$$\text{previous}[j,t] := \begin{cases} \text{parent}[j] & \text{if } t = \text{starttime}[j], \\ j & \text{otherwise.} \end{cases}$$

Using this specification and the data set in appendix B, AMPL generates a nonredundant MPS file.

4.2. TYPE 2: SCENARIO STRUCTURE WITH PERIOD-TO-PERIOD INDEPENDENCE OF DATA VALUE DISTRIBUTIONS

This type of problem uses random variable distributions for which data values can vary by time period but are independent of events in prior periods. The tree can

therefore be computed from specification of the distributions within each time period. (Problems with stationary distributions throughout all time periods are a special case, which could be computed from one set of distribution parameters alone.) This is an important consideration for the modeller, because it can greatly reduce the redundancy and hence the data requirements despite the obvious geometric growth in the number of nodes in the event tree. For example, consider the four-period tree of figure 2. In period 3 there are six nodes and in period 4 there are twelve, but data need only be specified for three and two branches, respectively (as demonstrated in table 1). All other parameter values can then be inferred by the program.

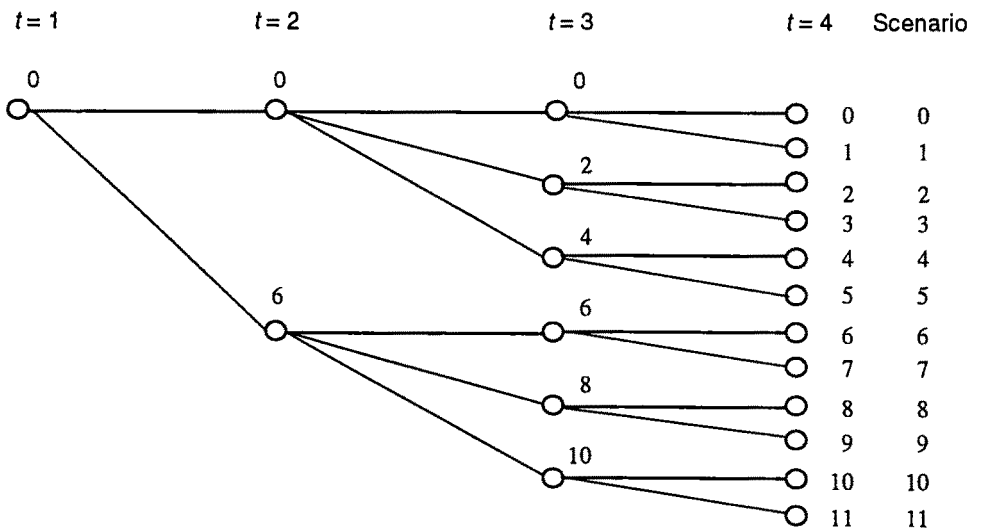


Figure 2. Period-to-period independent scenario tree for forestry application.

Table 1

Distribution of loss rates for forestry example.

Time	Branch	Loss rate	Probability
1	1	0.00	0.7
	2	0.20	0.3
2	1	0.00	0.4
	2	0.07	0.5
	3	0.20	0.1
3	1	0.04	0.5
	2	0.10	0.5

Appendix C gives an example of this problem type which arises in forest management. The objective is to determine a harvesting sequence which maximizes

the expected total value of wood harvested from a forest over a future time period with uncertain loss rates due to fires and other hazards. The forest is divided into age classes (areas holding trees of given ages); as trees mature, areas move from one age class to another, with new trees of zero age replacing trees that have been harvested. In addition, a fire destroys a random proportion of the forest left standing after harvesting. Trees remaining at the end of the planning period are valued and included as part of the objective.

Here we require analysis of the scenario structure which goes well beyond the algebraic formulation but which serves to greatly simplify the problem specification, compared to problems of type 1 above. Although the forestry problem's algebraic formulation merely assumes event sequences as in the problem above, the scenario tree underlying this problem can be completely specified by the number of time periods and the (unconditional) distribution of loss rates in each period, more precisely the number of possible realizations one is interested in. Parent and start time parameters can be calculated using elementary arithmetic operations if we utilize a top-to-bottom scenario labelling scheme. We illustrate this with the following example using the scenario tree given in figure 2.

This tree uses three loss distributions, described in table 1. These result in 12 scenarios, numbered sequentially from top to bottom starting at 0 (so as to simplify some of the arithmetic); we use the same convention as in the previous section regarding parent scenarios. The number above each node in the tree indicates the scenario in which data is specified for that node.

The following parameters then define the tree structure:

(a) The number of scenario paths coinciding with a node in period t. In scenarios of type 2, this number is constant across each period and can be denoted by branches[t]. This is of course just the product over the number of realizations in all time periods beyond t. In the tree of figure 2, branches[2] = 6, as each second-period node coincides with exactly six distinct scenario paths; similarly branches[1] = 12, branches[3] = 2, branches[4] = 1.

(b) The start time of a scenario j, calculated as follows:

$$\text{starttime}[j] = \min\{t \mid j \equiv 0 \pmod{\text{branches}[t]}\}.$$

(If there is no such t, then we will set starttime[j] = T. In figure 2 we have starttime[2] = 3, since $2 \equiv 0 \pmod{\text{branches}[3]}$ but $2 \not\equiv 0 \pmod{\text{branches}[2]}$.)

(c) The parent of a scenario j, which is 0 for the base scenario and otherwise is the largest integer multiple of the number of branches below the period prior to the start time of scenario j less than or equal to the scenario number. For example, the parent of scenario 4 is 0 since the start time of scenario 4 is $t = 3$ and there are 6 branches below period $t = 2$. The parent scenario can be computed in AMPL as $j - (j \bmod \text{branches}[\text{starttime}[j] - 1])$.

In the complete AMPL formulation (see appendix C) we define the scenario structure using the following:

(a) the parameter realizations: the number of realizations or possible outcomes in the random variable's distribution;
(b) the parameter loss_rate, indexed over time periods and realizations;
(c) the parameter cond_prob, indexed over time periods and realizations: the conditional probability of each outcome in the distribution;
(d) the parameter branches, indexed over time periods: the number of branches in the tree below a given time period;
(e) the index set scenarios, calculated as all integers between 1 and the number of branches below time period 1;
(f) the parameter starttime, indexed over scenarios, calculated as described above;
(g) the parameter parent indexed over scenarios computed as outlined above;
(h) the set event_nodes and the parameter previous, indexed over event_nodes as explained in the section on arbitrary fixed-horizon scenarios;
(i) the parameter rv_address, indexed over event_nodes, which associates with each node in the tree the number of the realization from the relevant (marginal) distribution and is defined by

```
rv_address[j,t] := 1 + (( j div branches[t]) mod realizations[t-1]);
```

(j) the parameter path_prob, indexed over event_nodes and defined by

```
path_prob[j,t]
    := product{tau in 2 . . t} cond_prob[tau-1, rv_address[j,tau]].
```

For example, in the tree of figure 2 the information for the node in period 3 of scenario 8 can be found using the second realization of the random variable since 1 + ((8 div branches[3]) mod realizations[2]) = 1 + ((8 div2) mod 3) = 2. The (partial) path probability represents the probability in period 1 of finding oneself on scenario j in period t.

Some of the indexing expressions are quite complicated and do not appear in the formulation in appendix C in quite this form, because the version of AMPL available to us at the time of writing did not support integer division.

4.3. TYPE 3: SCENARIOS WITH RANDOM WALK OR RANDOM WALK WITH PERIOD-INDEPENDENT DRIFT

This type of problem uses random variables for changes in scenario data whose distributions are independent from period to period. Realizations in each

scenario therefore depend on both the time period and prior history, although the increments can be specified by time period alone, regardless of scenario.

We illustrate this problem type with a modification of the MIDAS model of appendix B, using random walk distributions with drift to generate the rate movements. This does not change the algebraic formulation, since it does not explicitly contain the scenario structure; however, it does change the AMPL formulation, which is given in appendix D. The explicit scenario, parent and start time parameters used in our type 1 example are replaced by the parameters used in our type 2 illustration. In addition, drift parameters for service cost movements, exchange rate movements and interest earned on cash deposits are defined to give the period-to-period changes for each coefficient.

The final changes in this formulation are found in specification of the parameters affected by the changed distributions – debt service costs r, exchange rates ρ and the cash deposit interest rate i. These are calculated in the following manner:

$$r^k_{s,t}(e_j)_t = r^k_{s,t-1}(e_j)_{t-1} + \Delta r^k_{s,t}(e_{jt}),$$

$$\rho^k_t(e_j)_t = \rho^k_{t-1}(e_j)_{t-1} + \Delta\rho^k_t(e_{jt}),$$

$$i_t(e_j)_t = i_{t-1}(e_j)_{t-1} + \Delta i_t(e_{jt}).$$

The notation $\Delta r^k_{s,t}(e_{jt})$ suggests that the values taken by the increments are not dependent on what happened in the past, and that the number of realizations, the (conditional) probabilities and the values are the same no matter what prior history $(e_j)_{t-1}$ has been observed to date.

In AMPL, these parameter values are computed for all scenarios and time periods by calculating current tree positions and prior movements, using arithmetic operations similar to those found in our previous example.

The AMPL data set for a three-period problem with two-point distributions reflects these changes. Although it does not appear shorter than our explicit representation for such a small problem, it would be considerably shorter for larger, bushier trees in which the advantage of specifying a distribution once per time period instead of once per node would be apparent.

4.4. OTHER SCENARIO TYPES

We have identified at least two more scenario types: (4) scenario trees in which random variable distributions of increments depend on prior events (e.g. interest rate movements which are symmetrical for moderate interest levels but are skewed upwards if the levels are very low and skewed downwards if the levels are very high), and (5) scenario trees in which the number of decision variables depends on the current scenario. (The most obvious example of such a scenario tree is one in which

"coffin states" terminate paths of certain realizations.) The nonredundant, efficient formulation of these more complex scenario types is an interesting area for further work.

5. Concluding comments

We have shown through these examples that three common types of stochastic linear programs can be formulated in AMPL, using either parameterized specifications or explicit definitions of scenario structure. Viewing a scenario tree as one of branching event sequences, the scenario structure can be specified through scenario label, parent and start time parameters; redundancy can be eliminated by restricting time indexes on parameters, decision variables and constraints to times greater than scenario start times. Although this structure is not typically found in the algebraic formulations of SLP models, it is straightforward to implement as an extension of this formulation. In this manner we produced the full deterministic equivalent of our test problems, which AMPL then translated into standard MPS form. Moreover, we found AMPL's data consistency checking to be quite useful in identifying minor errors in our models and data.

The extent to which these results transfer to other algebraic modelling languages and LP formulation tools depends, of course, on the particular capabilities of the tools in question.

During this exercise, we identified several areas related to management of stochastic linear programs and other scenario-based models which bear further consideration. First, we discussed and tentatively discarded recommendations that AMPL provide automatic analysis of scenario structure so that scenario parameters could be calculated by high-level operators rather than explicitly specified by the modeller. This would bring the AMPL specification closer to the algebraic one but seems to pose problems in terms of linking data to scenarios and nodes when the structure is not explicitly stated beforehand. We would welcome comments on the extent to which AMPL might be modified in this direction.

Second, it would be useful to consider whether AMPL and similar tools should be extended to produce output usable by the more specialized solvers now being used to handle large SLPs. For example, the input format suggested by Birge et al. [1] extends the MPS form to produce a nonredundant set of files which explicitly specify the stochastic structure of these models so that they can be solved using decomposition algorithms. It would be very difficult indeed to extract this information from an MPS file without adding a specification of the tree structure and parameter dependencies.

Third, in our discussions we identified several "views" of scenarios, each of which appears useful in certain modelling situations. For instance, managers tend to view scenarios as branching paths, modellers often view them as event sequences and computational experts treat them as special cases of networks, with node-and-arc specifications. The extent to which these views can be interchanged by model manage-

ment systems and to which they can be used opportunistically to support model definition, manipulation and efficient solution is a fascinating topic for further work in management of scenario-based models.

Appendix A: A deterministic production problem (Fourer et al. [5])

ALGEBRAIC FORMULATION

Given $\mathcal{P}$ a set of products,
$\mathcal{R}$ a set of raw materials,
$T > 0$ the number of production periods,
$M > 0$ maximum total production per period;

and $a_{ij} \geq 0$ $i \in \mathcal{R}, j \in \mathcal{P}$: units of raw material i needed to manufacture one unit of product j;
$b_i \geq 0$ $i \in \mathcal{R}$: maximum initial stock of raw material i;
c_{jt} $j \in \mathcal{P}, t = 1,\ldots,T$: estimated profit (if ≥ 0) or disposal cost (if ≤ 0) of product j in period t;
$d_i \geq 0$ $i \in \mathcal{R}$: storage cost per period per unit of raw material i;
f_i $i \in \mathcal{R}$: estimated residual value (if ≥ 0) or disposal cost (if ≤ 0) of raw material after the last period.

Define $x_{jt} \geq 0$ $j \in \mathcal{P}$, $t = 1,\ldots,T$: units of product j manufactured in period t;
$s_{it} \geq 0$ $i \in \mathcal{R}$, $t = 1,\ldots,T+1$: units of raw material in storage at the beginning of period t.

Maximize $\sum_{t=1}^{T}(\sum_{j\in\mathcal{P}} c_{jt}x_{jt} - \sum_{i\in\mathcal{R}} d_i s_{it}) + \sum_{i\in\mathcal{R}} f_i s_{i,T+1}$:
total over all periods of estimated profit less storage cost, plus value of remaining raw materials over the last period;

subject to $\sum_{j\in\mathcal{P}} x_{jt} \leq M$,
$t = 1,\ldots T$: total production in period t must not exceed the specified maximum;

$s_{i1} \leq b_i$, $i \in \mathcal{R}$: units of raw material i on hand at the beginning of period 1 must not exceed the specified maximum;

$s_{i,t+1} = s_{it} - \sum_{j\in\mathcal{P}} a_{ij}x_{jt}$,
$i \in \mathcal{R}$, $t = 1,\ldots,T$: units of raw material i on hand at the beginning of period $t+1$ must equal units on hand at the beginning of period t less units used for production in period t.

AMPL MODEL FILE

```
### SETS ###
set prd;                        # products
set raw;                        # raw materials
```

```
### PARAMETERS  ###

param T > 0 integer;              # number of production periods

param max_prd > 0;                # maximum units of production per period

param units { raw,prd}  >= 0;     # units[i,j] is the quantity of raw material i
                                  # needed to manufacture one unit of product j

param init_stock { raw}  >= 0;    # init_stock[i] is the maximum initial stock
                                  # of raw material i

param profit { prd,1..T} ;        # profit[j,t] is the estimated value (if >= 0)
                                  # or disposal cost (if <= 0) of
                                  # a unit of product j in period t

param cost { raw}  >= 0;          # cost[i] is the storage cost
                                  # per unit per period of raw material i

param value { raw} ;              # value[i] is the estimated residual value
                                  # (if >= 0) or disposal cost (if <= 0)
                                  # of raw material i after the last period


### VARIABLES ###

var Make { prd,1..T} >= 0;        # Make[j,t] is the number of units of product j
                                  # manufactured in period t

var Store { raw,1..T+1} >= 0;     # Store[i,t] is the number of units of raw mate-
                                  # rial i in storage at the beginning of period t


### OBJECTIVE ###

maximize total_profit:

     sum { t in 1..T}  ( sum { j in prd}  profit[j,t] * Make[j,t] -
                       sum { i in raw}  cost[i] * Store[i,t] )

   + sum { i in raw}  value[i] * Store[i,T+1];

                  # Total over all periods of estimated profit,
                  # minus total over all periods of storage cost,
                  # plus value of remaining raw materials after last period
```

```
### CONSTRAINTS ###

subject to limit { t in 1..T} :  sum { j in prd}  Make[j,t] <= max_prd;
                  # Total production in each period must not exceed maximum

subject to start { i in raw} :  Store[i,1] <= init_stock[i];

                  # Units of each raw material in storage at beginning
                  # of period 1 must not exceed initial stock

subject to balance { i in raw, t in 1..T} :
     Store[i,t+1] = Store[i,t] - sum { j in prd}  units[i,j] * Make[j,t];

                  # Units of each raw material in storage
                  # at the beginning of any period t+1 must equal
                  # units in storage at the beginning of period t,
                  # less units used for production in period t
```

AMPL DATA FILE

```
data;
set prd := nuts bolts washers;
set raw := iron nickel;
param T := 4;
param max_prd := 123.7;

param units :       nuts    bolts    washers   :=
        iron        .79      .83       .92
        nickel      .21      .17       .08     ;

param profit :        1        2         3        4     :=
        nuts        1.73     1.8       1.6      2.2
        bolts       1.82     1.9       1.7      2.5
        washers     1.05     1.1        .95     1.33  ;

param :           init_stock    Cost    Value   :=
        iron        35.8         .03      .02
        nickel      7.32         .025    -.01    ;
end;
```

Appendix B: MIDAS stochastic programming problem

ALGEBRAIC FORMULATION

NOTATION:

s,t = $0,\ldots,T$ denote *time periods.*

T is the *length* of planning period or *horizon*.

$k = 1,\ldots,K$ denotes an available *debt type*.

$e_j := (e_{j1}, e_{j2},\ldots,e_{jT}), j = 1,\ldots,J$ denotes a *sequence* of (rate) *events* or *scenario*.

$(e_j)_t$ indicates that a variable or parameter is *contingent* on the *evolution* of event sequence e_j up to (and including) period t.

E_t denotes the set of all distinct event sequences $(e_{j1}, e_{j2},\ldots,e_{jt})$ which can occur up to and including period t. ($E_T = \{e_1,\ldots,e_J\}$ and E_1 is a singleton set.)

DECISION VARIABLES:

$B_t^k(e_j)_t$ dollar amount at par of debt type k *borrowed* in period t.

$O_{s,t}^k(e_j)_t$ dollar amount at par of debt type k borrowed in period s and *outstanding* at the end of period t.

$R_{s,t}^k(e_j)_t$ dollar amount at par of debt type k borrowed in period s and *retired* in period t.

$S_t(e_j)_t$ dollar value of *surplus* cash balance at the end of period t.

PARAMETERS:

$r_{s,t}^k(e_j)_t$ *service cost* in period t per dollar *outstanding* at the end of period $t-1$ of debt type k issued in period s.
(These parameters are used to handle interest payments and sinking fund contributions.)

$g_{s,t}^k(e_j)_t$ *cash outflows* per dollar for debt type k issued in period s, if *retired* during period t.
(These parameters are used to handle call premiums, sinking fund withdrawals and payments of accrued interest on retirements.)

$v_s^k(e_j)_T$ *market value* (in base currency) per dollar of debt of type k borrowed in period s and outstanding at the end of period T.

$\rho_t^k(e_j)_t$ *exchange rate* of foreign currency per unit of base currency appropriate to debt type k in period t.

$i_t(e_j)_t$ *interest* paid in period t per dollar of surplus cash balance at the end of period $t-1$.

$p(e_j)_T$ *probability* of event sequences e_j, $j = 1,\ldots,J$. ($\sum_{j=1}^{J} p(e_j)_T = 1$.)

f_t^k *issue costs* (excluding premium or discount) per dollar borrowed of debt type k issued in period t.

C_t *cash requirement* for period t. If negative, C_t indicates an *operating surplus*.

M_t *maximum* allowable *cash outflows* for debt service in period t.

N_t *maximum total borrowing* over all debt types in period t.

q_t^k *minimum borrowing* of debt type k in period t.

Q_t^k *maximum borrowing* of debt type k in period t.

L_t *minimum* dollar amount of debt (at par) *retired* in period t.

U_t *maximum* dollar amount of debt (at par) *retired* in period t.

$O^k_{0,0}$ *initial amount* of debt type k outstanding (borrowed before the start of the planning period).

S_0 *initial cash* surplus.

OBJECTIVE:

$$\min \sum_{j=1}^{J} p(e_j)_T \{\sum_{k=1}^{K} \sum_{t=0}^{T} v_t^k(e_j)_T O_{t,T}^k(e_j)_T - S_T(e_j)_T\}$$

(expected cost of retiring outstanding debt at *end* of period T).

CONSTRAINTS:

Cash requirements

For $(e_j)_t \in E_t$ and $t = 1,\ldots,T$

$$\begin{aligned} C_t = \sum_{k=1}^{K} \rho_t^k(e_j)_t \{&(1 - f_t^k) B_t^k(e_j)_t && \text{(net new borrowing)} \\ &- \sum_{s=0}^{t-1} [r_{s,t}^k(e_j)_t O_{s,t-1}^k(e_j)_{t-1} && \text{(interest on outstanding debt)} \\ &+ g_{s,t}^k(e_j)_t R_{s,t}^k(e_j)_t]\} && \text{(cash outflows on retirement)} \\ &+ S_{t-1}(e_j)_{t-1} && \text{(surplush cash in previous period)} \\ &+ i_t(e_j)_t S_{t-1}(e_j)_{t-1} && \text{(interest earned on surplus cash)} \\ &- S_t(e_j)_t && \text{(surplus cash at the end of current period)} \end{aligned}$$

Debt inventory by type

For $(e_j)_t \in E_t$, $s = 0,\ldots,t-1$, $t = 1,\ldots,T$ and $k = 1,\ldots,K$

$$\begin{aligned} O_{s,t}^k(e_j)_t - O_{s,t-1}^k(e_j)_{t-1} + R_{s,t}^k(e_j)_t &= 0 \\ O_{t,t}^k(e_j)_t - B_t^k(e_j)_t &= 0. \end{aligned}$$

Maximum cash outflows for debt service

For $(e_j)_t \in E_t$ and $t = 1,\ldots,T$

$$\sum_{k=1}^{K} \sum_{s=0}^{t-1} r_{s,t}^k(e_j)_t O_{s,t-1}^k(e_j)_{t-1} - i_t(e_j)_t S_{t-1}(e_j)_{t-1} \leq M_t.$$

Maximum total borrowing

For $(e_j)_t \in E_t$ and $t = 1,\ldots,T$

$$\sum_{k=1}^{K} \rho_t^k(e_j)_t B_t^k(e_j)_t \leq N_t.$$

Maximum debt issue size

For $(e_j)_t \in E_t$, $t = 1,\ldots,T$ and $k = 1,\ldots,K$

$$B_t^k(e_j)_t \leq Q_t^k.$$

Minimum debt issue size

For $(e_j)_t \in E_t$, $t = 1,\ldots,T$ and $k = 1,\ldots,K$

$$\text{either } B_t^k(e_j)_t = 0 \text{ or } B_t^k(e_j)_t \geq q_t^k (\geq 0).$$

Maturity smoothing

For $(e_j)_t \in E_t$ and $t = 1,\ldots,T$

$$L_t \leq \sum_{k=1}^{K} \sum_{s=0}^{t-1} R_{s,t}^{k}(e_j)_t \leq U_t.$$

Nonnegativity

For $(e_j)_t \in E_t$, $s = 0,\ldots,t-1$, $t = 1,\ldots,T$ and $k = 1,\ldots,K$

$$B_t^k(e_j)_t \geq 0, \quad O_{s,t}^k(e_j)_t \geq 0, \quad O_{t,t}^k(e_j)_t \geq 0, \quad R_{s,t}^k(e_j)_t \geq 0, \quad S_t(e_j)_t \geq 0.$$

AMPL MODEL FILE

```
###### MIDAS model file

param T > 0;            # number of time periods

### SETS ###

set periods := 0..T;
set debt_types;
set scenarios;

### PARAMETERS FOR SCENARIO STRUCTURE ###

param prob {scenarios};         # path probability
param parent {scenarios};       # parent scenario
param starttime {scenarios} > 0 integer;
                # first period in which the scenario
                # has a node with data different
                # from its parent scenario

### AUXILIARY SETS AND PARAMETERS ###

set event_nodes := {j in scenarios, t in 1..T: t >= starttime[j]};
set evolutions := {j in scenarios, s in 0..T, t in 1..T:
                s <= t and t >= starttime[j]};
param previous {(j,t) in event_nodes: t > 1} :=
   (if t = starttime[j] then parent[j] else j);

### OTHER PARAMETERS ###

param svc_cost {debt_types, evolutions} >= 0;
          # debt service cost per dollar outstanding;
          # includes interest per period and
          # sinking fund contributions
param ret_cost {debt_types, evolutions};
          # retirement discount/premium
param end_val {debt_types, scenarios, periods};
          # market value (in base currency)
          # per dollar of debt
```

```
param exch_rate{debt_types, event_nodes};
          # exchange rate of foreign currency
          # per unit of base currency
param cash_int {event_nodes} >=0;
          # interest on surplus cash
param issue_cost {debt_types,1..T}>=0;
param cash_req      {1..T};  # cash requirements
param max_cost      {1..T}>=0; # maximum debt cost per period
param max_Tbor      {1..T}>=0; # maximum total borrowing
param min_bor  {debt_types,1..T}>=0; # minimum borrowing of a debt type
param max_bor  {debt_types,1..T}>=0; # maximum borrowing of a debt type
param min_ret       {1..T}>=0; # minimum amount retired
param max_ret       {1..T}>=0; # maximum amount retired
param init_debt {debt_types}  >=0; # initial outstanding debt
param init_cash         >=0; # initial surplus cash

### VARIABLES ###

var borrow {debt_types, event_nodes}>= 0;
var outst {debt_types, evolutions}>= 0;
var retire {debt_types, evolutions}>= 0;
var delta {debt_types, event_nodes}binary; # These are used to code
                  # minimal and maximal borrowing
                  # but we would solve the LP
                  # relaxation instead.
var cash  {event_nodes} >= 0;

### OBJECTIVE ###

minimize endvalue:

sum {j in scenarios}
 prob[j] * ( sum {k in debt_types, s in 0..T} end_val[k,j,s] * outst [k,j,s,T]
       - cash [j,T]);

### CONSTRAINTS ###

 subject to balance {(j,t) in event_nodes}:

cash_req[t] = sum {k in debt_types}
   exch_rate[k,j,t] * ( (1-issue_cost[k,t])*borrow[k,j,t]
     - sum {s in 0..t-1} (svc_cost[k,j,s,t] *
        (if t = 1
        then init_debt[k]
        else outst[k,previous[j,t],s,t-1])
       + ret_cost[k,j,s,t]*retire[k,j,s,t] ))
      + ( if t = 1
       then init_cash
       else (1 + cash_int[j,t]) * cash[previous[j,t],t-1])
      - cash[j,t];

 subject to inventory {k in debt_types, (j,s,t) in evolutions}:
 outst[k,j,s,t] = (if s = t
       then borrow[k,j,s]
       else (if t = 1
         then init_debt[k]       - retire[k,j,s,t]
         else outst[k,previous[j,t],s,t-1] - retire[k,j,s,t]));
```

```
subject to debt_cost {(j,t) in event_nodes}:

   sum {k in debt_types, s in 0..t-1}
   (if t=1
    then svc_cost[k,j,s,t] * init_debt[k] - cash_int[j,t]*init_cash
    else svc_cost[k,j,s,t] * outst[k,previous[j,t],s,t-1]
       - cash_int[j,t] * cash[previous[j,t],t-1] )
   <= max_cost[t];

subject to market_max {(j,t) in event_nodes}:
   sum {k in debt_types} (exch_rate[k,j,t]*borrow[k,j,t]) <= max_Tbor[t];

subject to max_issue {k in debt_types, (j,t) in event_nodes}:
   borrow[k,j,t] - delta[k,j,t] * max_bor[k,t] <=0;

subject to min_issue {k in debt_types, (j,t) in event_nodes}:
   borrow[k,j,t] - delta[k,j,t] * min_bor[k,t] >=0;

subject to mat_smooth {(j,t) in event_nodes}:
   min_ret[t] <= sum {k in debt_types, s in 0..t-1} retire[k,j,s,t]
                 <= max_ret[t];
```

AMPL DATA FILE

```
data;

set debt_types := can5yr us5yr ;
set scenarios := 1 2 ;

param T := 3;          # time interval in years

param :   prob   parent   starttime :=
   1   .75    0      1
   2   .25    1      3 ;

param svc_cost default 0.0 :=

[can5yr,1,*,*]   0    1    .11 # scenario 1: falling rates
         0    2   .11
         1    2   .105
         0    3   .11
         1    3   .105
         2    3   .10
         3    3   .095

[can5yr,2,*,*]   0    3    .11 # scenario 2: rise at end
         1    3   .105
         2    3   .10
         3    3   .105

[us5yr,1,*,*]    0    1    .095
         0    2   .095
         1    2   .091
         0    3   .095
         1    3   .091
         2    3   .087
         3    3   .083
```

```
[us5yr,2,*,*]    0    3    .095
        1    3    .091
        2    3    .087
        3    3    .091 ;

param issue_cost (tr) :   can5yr  us5yr :=
        1    .009    .015
        2    .009    .015
        3    .009    .015 ;

param ret_cost default 99.9 :=

[can5yr,1,*,*]    0   3   1.0

[can5yr,2,*,*]    0   3   1.0

[us5yr,1,*,*]     0   2   1.0 ;

param end_val default 0.0 :=

[can5yr,1,*]      1       1.2
        2       1.1
        3       1.05

[can5yr,2,*]      1       .95
        2       .9
        3       .95

[us5yr,1,*]       1       1.2
        2       1.1
        3       1.05

[us5yr,2,*]       1       1.03
        2       .95
        3       .8 ;

param exch_rate default 1.0 :=

[us5yr,1,*]       1       1.18
        2       1.21
        3       1.25

[us5yr,2,*]       3       1.25;

param cash_int default 0.0 :=

[1,*]         1    .08
        2    .075
        3    .07
[2,*]         3    .0735 ;

param :    cash_req max_cost  max_ret  min_ret max_Tbor :=
   1   217.0   75.0  200.0   0.0  500.0
   2   296.0  100.0  200.0   0.0  500.0
   3   310.0  150.0  200.0   0.0  500.0;

param max_bor (tr) :    can5yr   us5yr :=
        1   250.0   250.0
        2   250.0   250.0
        3   250.0   250.0 ;
```

```
param min_bor default 0.0 (tr) :
          can5yr   us5yr :=
       1     .      .
       2     .      .
       3     .      .  ;

param init_debt :=   can5yr 200.0   us5yr 100.0 ;

param init_cash := 50.0 ;
end;
```

Appendix C: Stochastic forestry model

ALGEBRAIC FORMULATION

NOTATION:

$t = 1,\ldots,T$ denote *time periods*.

T is the *length* of planning period or *horizon*.

$k = 1,\ldots,K$ denotes an *age class*.

$e_j := (e_{j1}, e_{j2},\ldots,e_{jT}), j = 1,\ldots,J$ denotes a *sequence* of (rate) *events* or *scenario*.

$(e_j)_t$ indicates that a variable or parameter is *contingent* on the *evolution* of event sequence e_j up to (and including) period t.

E_t denotes the set of all distinct event sequences $(e_{j1}, e_{j2},\ldots,e_{jt})$ which can occur up to and including period t. ($E_T = \{e_1,\ldots,e_J\}$ and E_1 is a singleton set.)

DECISION VARIABLES:

$s_t^k(e_j)_t$ area having trees in age class k at beginning of period t.

$h_t^k(e_j)_t$ area of age class k harvested during period t.

PARAMETERS:

$f_t^{k,l}(e_j)_t$ *proportion* of age class k in period t that moves to age class l by the beginning of period $t+1$.

y_k *yield* per unit area of trees in age class k. This is time-invariant.

v_k *value* per unit area of trees in age class k left standing at the time horizon.

δ *discount* factor.

α, β maximal allowable change in harvest volume from one period to the next.

s_1^k initial area having trees in age class k.

$p_t(e_j)_t$ probability of having realized event sequence $(e_j)_t$.

OBJECTIVE:

$$\max \sum_{t=1}^{T} \sum_{j \in E_t} \delta^t p_t(e_j)_t \sum_{k=1}^{K} y_k h_t^k(e_j)_t \qquad \text{(expected harvest in period } t\text{)}$$

$$+ \sum_{j \in E_T} \delta^{T+1} p_T(e_j)_T \sum_{k=1}^{K} v_k s_{T+1}^k(e_j)_t \qquad \text{(expected value of forest left standing at horizon).}$$

CONSTRAINTS:

AVAILABLILITY:

For $k = 1,\ldots,K$, $t = 1,\ldots T$, $j \in E_t$,

$$h_t^k(e_j)_t - s_t^k(e_j)_t \leq 0.$$

INVENTORY:

For $k = 2,\ldots,K-1$, $t = 2,\ldots T+1$, $j \in E_t$,

$$s_t^k(e_j)_t = f_{t-1}^{k-1,k}(e_j)_{t-1}[s_{t-1}^{k-1}(e_j)_{t-1} - h_{t-1}^{k-1}(e_j)_{t-1}].$$

For $t = 2,\ldots,T+1$, $j \in E_t$,

$$s_t^1(e_j)_t = \sum_{k=1}^{K} f_{t-1}^{k-1,k}(e_j)_{t-1}[s_{t-1}^{k-1}(e_j)_{t-1} - h_{t-1}^{k-1}(e_j)_{t-1}],$$

$$s_t^K(e_j)_t = f_{t-1}^{K-1,K}(e_j)_{t-1}[s_{t-1}^{K-1}(e_j)_{t-1} - h_{t-1}^{K-1}(e_j)_{t-1}]$$

$$+ f_{t-1}^{K,K}(e_j)_{t-1}[s_{t-1}^{K}(e_j)_{t-1} - h_{t-1}^{K}(e_j)_{t-1}].$$

HARVEST FLOW:

For $k = 1,\ldots,T$, $t = 2,\ldots T$, $j \in E_t$,

$$\alpha \sum_{k=1}^{K} y_k h_{t-1}^k(e_j)_{t-1} - \sum_{k=1}^{K} y_k h_t^k(e_j)_t \leq 0,$$

$$\beta \sum_{k=1}^{K} y_k h_{t-1}^k(e_j)_{t-1} - \sum_{k=1}^{K} y_k h_t^k(e_j)_t \geq 0.$$

NONNEGATIVITY:

For $t = 1,\ldots,T$, $k = 1,\ldots,K$, $j \in E_t$,

$$s_{t+1}^k(e_j)_t \geq 0 \qquad h_t^k(e_j)_t \geq 0.$$

AMPL MODEL FILE

```
### Forestry model
### General (deterministic) parameters

param T > 0;        # number of time periods
param K > 0;        # number of age classes

set ageclasses := 1..K;
set periods  := 1..T;

param yield {ageclasses} >= 0;       # yield per acre
param presval {ageclasses} >= 0;       # value per acre at horizon
param state1 {ageclasses} >= 0;       # initial stand of timber
param discount >= 0;           # discount factor
param flow_min >= 0;
param flow_max >= 0;           # used for flow constraints
param mean_loss_rate <= 1;

### Parameters to define the random structure

param realizations {periods} > 0 integer;
param loss_rate {t in periods, s in 1..realizations[t]} >= 0;
param cond_prob {t in periods, s in 1..realizations[t]} <= 1;

### Auxiliary sets and parameters for easier manipulation of scenarios

param branches {t in 1..T+1} :=
      (if t <= T then prod {s in t..T} realizations[s]
          else 1);
set  scenarios    := 0..(prod {t in periods} realizations[t] - 1);

param starttime {j in scenarios} :=
        min {t in 1..T+1: (j mod branches[t]) = 0} t;

param parent  {j in scenarios} :=
        (if j = 0 then 0
           else j - (j mod branches[starttime[j]-1]));

set  event_nodes := {j in scenarios, t in 1..T+1: t >= starttime[j]};

param rv_address {(j,t) in event_nodes: t > 1} :=
  ((( j - (j mod branches[t])) / branches[t]) mod realizations[t-1]) + 1;

param previous {(j,t) in event_nodes: t > 1} :=
  ( if t = starttime[j] then parent[j] else j );

param path_prob {(j,t) in event_nodes} :=
   ( if t = 1 then 1
      else path_prob[previous[j,t],t-1] *
        cond_prob[t-1,rv_address[j,t]]);

### VARIABLES

var state {ageclasses,t in 1..T+1, s in scenarios: starttime[s] <= t} >= 0;
var harvest {ageclasses,t in periods, s in scenarios: starttime[s] <= t} >= 0;
```

```
### Objective:

maximize total_yield:
sum {k in ageclasses, (s,t) in event_nodes}
  ( if t <= T
   then (discount^t * yield[k] * harvest[k,t,s] * path_prob[s,t])
   else (discount^t * presval[k] * state[k,t,s] * path_prob[s,t]) );

### Constraints:

subject to avail 'availability of timber'
  {k in ageclasses, (s,t) in event_nodes: t <= T }:
   harvest[k,t,s] - state[k,t,s] <= 0;

subject to inventory
  {k in ageclasses, (s,t) in event_nodes}:
  state[k,t,s] =
  (if t = 1
  then state1[k]
  else (if k = 1
   then sum {a in ageclasses}
     (loss_rate[t-1,rv_address[s,t]] * state[a,t-1,previous[s,t]] +
    (1-loss_rate[t-1,rv_address[s,t]])* harvest[a,t-1,previous[s,t]])
  else (1-loss_rate[t-1,rv_address[s,t]]) *
     ( state[k-1,t-1,previous[s,t]] - harvest[k-1,t-1,previous[s,t]]
    + (if k = K then
      state[k, t-1,previous[s,t]] - harvest[k, t-1,previous[s,t]]))));

subject to final_inventory
  {k in ageclasses, s in scenarios}:
  state[k,T+1,s] =
  (if k = 1
   then sum {a in ageclasses}
     (mean_loss_rate * state[a,T,previous[s,T+1]] +
    (1-mean_loss_rate)* harvest[a,T,previous[s,T+1]])
  else (1-mean_loss_rate) *
     ( state[k-1,T,previous[s,T+1]] - harvest[k-1,T,previous[s,T+1]]
    + (if k = K then
      state[k, T,previous[s,T+1]] - harvest[k, T,previous[s,T+1]])));

subject to flow1 {t in 2..T, s in scenarios: starttime[s] <= t}:
  (1-flow_min)*sum {k in ageclasses}
    yield[k]*harvest[k,t-1,previous[s,t]] -
   sum {k in ageclasses} yield[k]*harvest[k,t,s]  <= 0;

subject to flow2 {t in 2..T, s in scenarios: starttime[s] <= t}:
  (1+flow_max)*sum {k in ageclasses}
    yield[k]*harvest[k,t-1,previous[s,t]] -
   sum {k in ageclasses} yield[k]*harvest[k,t,s]  >= 0;
```

AMPL DATA FILE

```
data;

param T := 3;
param K := 4;

param realizations :=
   1  2
   2  3
   3  2 ;

param : yield presval  state1 :=
 1    0   330   0.366
 2    62   400   3.408
 3   246   520   26.153
 4   302   590   64.810 ;

param discount := 0.9;
param flow_min := 0.1;
param flow_max := 0.1;
param mean_loss_rate := 0.07;

param loss_rate :=
 [1,*] 1  0.0
    2  0.2
 [2,*] 1  0.0
    2  0.07
    3  0.2
 [3,*] 1  0.04
    2  0.1 ;

param cond_prob :=
 [1,*] 1  0.7
    2  0.3
 [2,*] 1  0.4
    2  0.5
    3  0.1
 [3,*] 1  0.5
    2  0.5 ;

end;
```

Appendix D: Random walk scenario model

(For algebraic formulation, see appendix B)

AMPL MODEL FILE

```
param T > 0;          # number of time periods

### SETS ###

set periods := 0..T;
set debt_types;
```

```
### Parameters to define the random structure
param realizations {t in 1..T} > 0 integer;
param cond_prob { t in 1..T, s in 1..realizations[t]} <= 1;

### Auxiliary sets and parameters for easier manipulation of scenarios
param branches {t in 1..T}:= prod {s in t..T} realizations[s];

set scenarios    := 0..(prod {t in 1..T} realizations[t] - 1);
param starttime {j in scenarios} :=
        min {t in 1..T: (j mod branches[t]) = 0} t;
param parent {j in scenarios} :=
        (if j = 0 then 0
           else j - (j mod branches[starttime[j]-1]));
set event_nodes := {j in scenarios, t in periods: t >= starttime[j]};
set evolutions := {j in scenarios, s in 0..T, t in 1..T:
          s <= t and t >= starttime[j]};
param previous {(j,t) in event_nodes: t > 1} :=
   (if t = starttime[j] then parent[j] else j);
param rv_address {(j,t) in event_nodes: t > 1} :=
 ((( j - (j mod branches[t])) / branches[t]) mod realizations[t-1]) + 1;
param path_prob {(j,t) in event_nodes}:=
   ( if t = 1 then 1
       else path_prob[previous[j,t],t-1] *
        cond_prob[t-1,rv_address[j,t]] );
param prob {j in scenarios} := path_prob[j,T];


### PARAMETERS ###

param issue_cost {debt_types,1..T}>=0;
param cash_req     {1..T};  # cash requirements
param max_cost     {1..T}>=0;  # maximum debt cost per period
param max_Tbor     {1..T}>=0;  # maximum total borrowing
param min_bor {debt_types,1..T}>=0;  # minimum borrowing of a debt type
param max_bor {debt_types,1..T}>=0;  # maximum borrowing of a debt type
param min_ret      {1..T}>=0;  # minimum amount retired
param max_ret      {1..T}>=0;  # maximum amount retired
param init_debt {debt_types}  >=0;  # initial outstanding debt
param init_cash         >=0;  # initial surplus cash
param init_int {debt_types}  >=0;  # initial interest rates (period 1)
param prev_int {debt_types}  >=0;  # prior interest rates (period 0)
param init_cr          >=0;  # initial rate for cash
param init_exc {debt_types}  >=0;  # initial exchange rate

### The next set of parameters is used to define the drifts
param int_drift {debt_types, t in 1..T, s in 1..realizations[t]};
param exc_drift {debt_types, t in 1..T, s in 1..realizations[t]};
param cash_drift     {t in 1..T, s in 1..realizations[t]};

# Now the rate parameters are defined in terms of starting values and drift
param interest {k in debt_types, (j,s,t) in evolutions} :=
  ( if s = 0 then prev_int[k]
      else (if s = 1 then init_int[k]
      else interest[k,previous[j,t],s-1,t-1] +
       int_drift[k,t-1,rv_address[j,t]] ));
```

```
                # interest per dollar
                # outstanding per period AND
                # sinking fund contributions
param exch_rate{k in debt_types, (j,t) in event_nodes} :=
  ( if t = 1 then init_exc[k]
      else exch_rate[k,previous[j,t],t-1] +
       exc_drift[k,t-1,rv_address[j,t]] );
                # exchange rate of foreign currency
                # per unit of base currency
param cash_int {(j,t) in event_nodes} :=
  ( if t = 1 then init_cr
      else cash_int[previous[j,t],t-1] +
       cash_drift[t-1,rv_address[j,t]] );
                # interest on surplus cash

### The next set of parameters should be stochastic, but they definitely do
### not exhibit the same drift. In fact, they should be computed from other
### data using NPV calculations.
### It seems sensible to pretend they are deterministic in this test...
param ret_cost {debt_types, s in 0..T, t in 1..T: s < t};
                # retirement discount/premium

param end_val {debt_types, s in 0..T};
                # market value (in base currency)
                # per dollar of debt

### VARIABLES ###

var borrow {debt_types, (j,t) in event_nodes}>=0;
var outst {debt_types, (j,s,t) in evolutions}>=0;
var retire {debt_types, (j,s,t) in evolutions}>=0;
var cash      {(j,t) in event_nodes}>=0;
var delta {debt_types, (j,t) in event_nodes}binary;

### OBJECTIVE ###

minimize endvalue:

sum {j in scenarios}
  prob[j] * ( sum {k in debt_types, t in 0..T}
    end_val[k,t] * outst [k,j,t,T] - cash [j,T]);

### CONSTRAINTS ###

subject to balance {(j,t) in event_nodes}:

cash_req[t] = sum {k in debt_types}
   exch_rate[k,j,t] * ( (1-issue_cost[k,t])*borrow[k,j,t]
      - sum {s in 0..t-1} ( interest[k,j,s,t] *
         (if t = 1
          then init_debt[k]
          else outst[k,previous[j,t],s,t-1])
        + ret_cost[k,s,t]*retire[k,j,s,t] ))
     + ( if t = 1
        then init_cash
        else (1 + cash_int[j,t]) * cash[previous[j,t],t-1])
     - cash[j,t];
```

```
subject to inventory {k in debt_types, (j,s,t) in evolutions}:
outst[k,j,s,t] = (if s = t
        then borrow[k,j,s]
        else (if t = 1
          then init_debt[k]
          else outst[k,previous[j,t],s,t-1])
                - retire[k,j,s,t]);

subject to debt_cost {(j,t) in event_nodes}:

sum {k in debt_types, s in 0..t-1}
   (if t=1
    then interest[k,j,s,t] * init_debt[k] - cash_int[j,t]*init_cash
    else interest[k,j,s,t] * outst[k,previous[j,t],s,t-1]
         - cash_int[j,t] * cash[previous[j,t],t-1])
   <= max_cost[t];

subject to market_max {(j,t) in event_nodes}:
   sum {k in debt_types} (exch_rate[k,j,t]*borrow[k,j,t]) <= max_Tbor[t];

subject to max_issue {k in debt_types, (j,t) in event_nodes}:
   borrow[k,j,t] - delta[k,j,t] * max_bor[k,t] <=0;

subject to min_issue {k in debt_types, (j,t) in event_nodes}:
   borrow[k,j,t] - delta[k,j,t] * min_bor[k,t] >=0;

subject to delta_value{k in debt_types, (j,t) in event_nodes}:
   0 <= delta[k,j,t] <= 1;

subject to mat_smooth {(j,t) in event_nodes}:
   min_ret[t] <= sum {k in debt_types, s in 0..t-1} retire[k,j,s,t]
                      <= max_ret[t];
```

AMPL DATA FILE

```
set debt_types := can5yr us5yr ;

param T := 3;
# time interval is years

param realizations :=
    1  2
    2  2
    3  1 ;

param cond_prob :=
  [1,*] 1  0.6
     2  0.4
  [2,*] 1  0.6
     2  0.4
  [3,*] 1  1.0 ;

param prev_int default 0.0 :=
[can5yr]  .11
[us5yr]  .095;
```

```
param init_int default 0.0 :=
[can5yr] .105
[us5yr] .091;

param init_exc default 1.0 := [us5yr] 1.15;

param init_cr := .03;

param issue_cost (tr) :   can5yr   us5yr :=
        1   .009     .015
        2   .009     .015
        3   .009     .015 ;

param ret_cost default 99.9 :=

[can5yr,*,*]    0   3   1.0

[us5yr,*,*]     0   2   1.0 ;

param int_drift :=
[can5yr,*,*]    1  1  0.005
        1  2 -0.001
        2  1  0.005
        2  2 -0.001
        3  1  0.002

[us5yr,*,*]     1  1  0.004
        1  2  0.002
        2  1  0.006
        2  2  0.000
        3  1  0.003 ;
param exc_drift default 0.0 :=
[us5yr,*,*]     1  1  0.02
        1  2 -0.02
        2  1  0.02
        2  2 -0.02
        3  1  0.005 ;

param cash_drift default 0.0 :=
[1,*]       1  0.005
        2  0.0
[2,*]       1  0.005
        2  0.0
[3,*]       1  0.0 ;

param end_val default 0.0 :=

[can5yr,*]     1      1.2
        2      1.1
        3      1.05

[us5yr,*]      1      1.2
        2      1.1
        3      1.05;

param :   cash_req max_cost  max_ret min_ret max_Tbor :=
   1   217.0   75.0  200.0   0.0  500.0
   2   296.0  100.0  200.0   0.0  500.0
   3   310.0  150.0  200.0   0.0  500.0;
```

```
param max_bor (tr) :   can5yr   us5yr :=
       1   250.0   250.0
       2   250.0   250.0
       3   250.0   250.0 ;

param min_bor default 0.0 (tr) :
          can5yr   us5yr :=
       1     .     .
       2     .     .
       3     .     .  ;

param init_debt :=   can5yr 200.0   us5yr 100.0 ;

param init_cash := 50.0 ;
end;
```

Acknowledgements

We gratefully acknowledge the support for this research by the Natural Sciences and Engineering Research Council of Canada and by the Social Sciences and Humanities Research Council of Canada. We thank two anonymous referees and the guest editor for careful reading of the paper and the AMPL model files.

References

[1] J.R. Birge, M.A.H. Dempster, H.I. Gassmann, E.A. Gunn, A.J. King and S.W. Wallace, A standard input format for stochastic linear programs, COAL Newsletter 17(1987)1–20.

[2] J. Bisschop and A. Meeraus, On the development of a general algebraic modelling system in a strategic environment, Math. Progr. Study 20(1982)1–29.

[3] A. Brooke, D. Kendrick and A. Meeraus, *GAMS: A User's Guide*, 2nd ed. (Scientific Press, South San Francisco, CA, 1992).

[4] R. Fourer, New directions for algebraic modelling languages, Paper presented at the *IMPS Roundtable*, University of Colorado at Denver (April, 1991).

[5] R. Fourer, D.M. Gay and B.W. Kernighan, *AMPL: A Modeling Language for Mathematical Programming* (Scientific Press, South San Francisco, CA, 1993).

[6] H.I. Gassmann, MSLiP: A computer code for the multistage stochastic linear programming problem, Math. Progr. 47(1990)407–423.

[7] H.J. Greenberg, A bibliography for the development of an intelligent mathematical programming system, Technical Report, University of Colorado at Denver (April, 1991).

[8] A.M. Ireland, An intelligent decision suport system for debt management, unpublished Ph.D. dissertation, Dalhousie University, Halifax, 1990.

[9] M. Lane and P. Hutchinson, A model for managing a certificate of deposit portfolio under uncertainty, in: *Stochastic Programming*, ed. M.A.H. Dempster (Academic Press, London, 1980).

[10] I.J. Lustig, J.M. Mulvey and T.J. Carpenter, Formulating two-stage stochastic programs for interior point methods, Oper. Res. 39(1991)757–770.