Automatic Formulation of Stochastic Programs Via an Algebraic Modeling Language

THENIE, Julien, VAN DELFT, Christian, VIAL, Jean-Philippe

DOI : 10.1007/s10287-006-0022-z

CMS (2007) 4:17–40
DOI 10.1007/s10287-006-0022-z

ORIGINAL PAPER

Automatic formulation of stochastic programs via an algebraic modeling language

J. Thénié · Ch. van Delft · J.-Ph.Vial

Published online: 17 June 2006

Abstract This paper presents an open source tool that automatically generates the so-called deterministic equivalent in stochastic programming. The tool is based on the algebraic modeling language AMPL. The user is only required to provide the deterministic version of the stochastic problem and the information on the stochastic process, either as scenarios or as a transitions-based event tree.

Keywords Stochastic programming · Algebraic modeling language · Deterministic equivalent

MSC code 90C15 Stochastic programming

1 Introduction

Sequential decision-making under uncertainty is a common situation that decision-makers face. Stochastic programming is a powerful analytical tool to support this process. As a scientific field stochastic programming has been in existence for many years and has received many contributions (see, for example Birge and Louveaux (1997), Kall and Wallace (1994) and the website "stoprog.org" for exhaustive references). Yet this tool is not widely used, mainly because users find it difficult to handle and implement. Indeed, the complexity

J. Thénié (✉) · J.-Ph.Vial
HEC/Logilab/Department of Management Studies,
University of Geneva, 40 Bd du pont d'Arve,
1211 Geneva 4, Switzerland
e-mail: julien.thenie@hec.unige.ch

Ch. van Delft
HEC School of Management,
Paris (GREGHEC),
78 351 Jouy-en-Josas Cedex, France

of implementing stochastic programming is considerably higher than in the deterministic case. For the latter, algebraic modeling languages, in short AML, make it possible for users who are not experts in mathematical programming to formulate and develop complex problems rapidly and have them solved.

A multistage stochastic programming problem can be interpreted as the combination of a multistage dynamic model and a discrete time stochastic process. If the stochastic process has finitely many states, one can formulate the stochastic program as a deterministic problem, the so-called *deterministic equivalent*. To build this model, one has to describe the two base components separately – the deterministic multistage dynamic model and the stochastic process – and then define variables and constraints contingent on the realization of the stochastic process. This last operation is tedious, if not difficult, because the contingent variables and constraints are in great number and must be appropriately hooked to specific events. The development of automatic procedures to build this model is an active research stream in stochastic programming (Condevaux-Lanloy et al. 2002, 2004; Fourer and Gay 1997, 2004; Valente et al. 2001). In any case, the dimension of the deterministic equivalent is a major issue in stochastic programming. Devising efficient solution methods is still an open field. It is thus important to give the user the opportunity to experiment with solution methods of his choice.

The present work achieves a twofold goal. First, we derive rules to generate event trees in a way similar to (van Delft and Vial 2004) for the transition-based and the scenario-based description. Second, we present a script that automatically generates the deterministic equivalent. The script is generic in that it applies to any problem, provided the user complies with simple formulation rules in the writing of the base deterministic model and the event tree description. In our approach we use AMPL (Fourer et al. 1993), which is one of the available commercial AML. The extension to another AML is conceivable if the language is endowed with analogous recursive definitions properties.

In Condevaux-Lanloy et al. (2002), Dormer et al. (2005) and IBM (1998) the authors propose a black-box approach, whose input consists in two files that are provided by the user. The output is the numerical solution of the deterministic equivalent. The input files describe the deterministic information and the stochastic part. Data are to be transmitted via SMPS (Edwards et al. 1985; Gassmann and Schweitzer 2001), an extension of the MPS format (IBM 1998; Murtagh 1981). For practical applications, even if some AML is used to build the data, this approach is not as straightforward as it seems: at first, the whole procedure is likely to appear quite tedious and long for practitioners and furthermore, some coefficients in the deterministic equivalent may be given as complex functions of deterministic and stochastic data, which cannot be easily translated as such in SMPS file. Furthermore, the control the user has on his model is through data transmission, with a model building process which is externally defined. Basically, in this approach the model building process is not really under direct control of the user. For example, the user cannot implement specific resolution algorithm via specific packages or decomposition methods.

Another approach, the one we choose to develop here, consists of exploiting the AML features in order to formulate the full deterministic equivalent, see van Deft and Vial (2004), and having a direct vision and control of this model. Then the user sends the problem to a commercial solver, knowing that nowadays such solvers can often handle large linear programming problems. Such an approach, which permits the user to develop the model, is a necessary condition during the modeling process. As a matter of fact, in real-life application the development of a model of the considered problem is a complex issue involving the setting of critical assumptions and approximations (as the choice of the parameters of influence, time horizon, randomness modeling, ...). It is thus an advantage for the user to control and understand the model structure during this development process.

In the hope of narrowing the gap between users and the available modeling and algorithmic tools, we propose some simple techniques to convert multistage deterministic problems into stochastic programming ones. In this framework, the deterministic part is entered as a standard AMPL model. The stochastic component is described via a collection of simple recursive formulas that allow AMPL[1] to generate the event tree for the stochastic process outcomes and compute the associated probabilities. We show that, in our framework, putting together the deterministic programming problem and the stochastic information can be made automatic by an appropriate script. We give the rules to write the script in terms of elementary operations. With our approach, the user is just asked to build the deterministic model and to provide the numerical data relative to the stochastic process. The script produces two files, a model file consisting of the full deterministic equivalent, and a data file. With those two files, AMPL can build the model and the instance to be passed to the solver that the user wishes to use.

Our methodology is inspired by the principles exposed in Gassman and Ireland (1996). We believe our proposal meets the main objectives of Gassman and Ireland (1996) and offers the user open access and fulls control of the deterministic equivalent. Our approach extends some of the ideas that appeared in Fragnière et al. (2000). There, the authors were able to show that a portfolio selection model with a million scenario could be generated by an AML and the data processed by a structure-exploiting tool called SET (Fragnière et al. 2000a) to feed general purpose optimization codes for decomposition. More recently, a similar scheme has been applied to the analysis of a supply chain management problem (van Delft and Vial 2004). In both cases, the underlying stochastic process is represented by an event tree and a one-step transition process. However, in many stochastic programming applications the stochastic process is described via scenarios. We thus devised an extension to that case. The originality of our approach is that it is entirely based on available functionalities of AMPL (Fourer et al. 1993) and GNU MATHPROG (Makhorin 2005), a free subset of AMPL. Our tool can be freely duplicated.

[1] In theory, the approach can also be extended to any AML that supports recursive definitions.

This paper is organized as follows. Section 2 presents the basic assumptions which characterize the stochastic optimization models considered in this paper. Section 3 deals with the modeling of the event tree. Section 4 presents a simple example to illustrate the concepts previously introduced. It will also be used in the later sections. Section 5 defines the primitives that make it possible to navigate through the tree. In section 6, we discuss the automatic merging of the linear deterministic optimization problem and the stochastic information embedded in the event tree to build the so-called deterministic equivalent, and we provide an example. In the conclusion, we summarize our contributions.

2 Stochastic programming formulation

Consider the following deterministic dynamic problem

$$\begin{aligned} \min \quad & h_0(x_0) + \sum_{t=1}^{\mathrm{T}} h_t(X_t, \Xi_t) && \text{(1a)} \\ \text{s.t.} \quad & f_t(X_t, \Xi_t) = 0, \qquad t = 1, \dots, T, && \text{(1b)} \\ & f_0(x_0) = 0, && \text{(1c)} \\ & g_t(X_t, \Xi_t) \leq 0, \qquad t = 1, \dots, T, && \text{(1d)} \\ & g_0(x_0) \leq 0, && \text{(1e)} \end{aligned}$$

where $X_t = (x_0, x_1, \dots x_t)$, with $x_t \in R^{p_t}$, is a decision variable and $\Xi_t = (\xi_1, \dots \xi_t)$, with $\xi_t \in R^{q_t}$, is a parameter. The equality constraints (1b) and (1c) are usually associated with the dynamics of the problem. In general, f_t, f_0, g_t and g_0 are vector-valued functions. Some problems may involve decision variables with delayed effect. To account for the time lag, one must use doubly indexed variables, e.g. $x_{t,t+k}$, to express the fact that the decision is taken at time t and takes effect at time $t + k$. For the sake of simplicity, we do not consider this notational extension in the main text. For more details, see subsection 6.4 or (van Delft and Vial 2004).

A stochastic version of Problem (1) is as follows

$$\begin{aligned} \min \quad & h_0(x_0) + E_\xi \left\{ \sum_{t=1}^{\mathrm{T}} h_t(X_t, \Xi_t) \right\} && \text{(2a)} \\ \text{s.t.} \quad & f_t(X_t, \Xi_t) = 0, \qquad t = 1, \dots, T, && \text{(2b)} \\ & f_0(x_0) = 0, && \text{(2c)} \\ & g_t(X_t, \Xi_t) \leq 0, \qquad t = 1, \dots, T, && \text{(2d)} \\ & g_0(x_0) \leq 0, && \text{(2e)} \end{aligned}$$

where, in this stochastic counterpart, the parameter Ξ_t is a stochastic process. In that general framework, the constraints are required to hold in an almost sure sense.

In the sequel, we shall use the concept of scenario.

Definition 1. *Each scenario ω is a full realization of the random process Ξ_T, namely*

$$\Xi_T(\omega) = (\xi_1(\omega), \ldots, \xi_T(\omega)).$$

Similarly,

$$\Xi_t(\omega) = (\xi_1(\omega), \ldots, \xi_t(\omega)).$$

When the number of scenarios is finite, Problem (2) becomes a regular finite-dimensional mathematical programming problem. Thus, we posit the assumption:

Assumption 1. *We assume that the process Ξ_T has a finite number N of scenarios $\omega \in \Omega = \{\omega_1, \ldots, \omega_N\}$. Scenario ω has an occurrence probability π_ω, with $\sum_{\omega\in\Omega} \pi_\omega = 1$. Furthermore, this stochastic process Ξ_T is independent of the decision vectors X_t.*

It can be seen that in Assumption 1 the scenarios can be identified by arbitrary alphanumeric codes. For the sake of simpler notation, one often chooses $\omega_i = i$ and this is what we use in the sequel. Nevertheless, for clarity purposes, we shall sometimes keep with the more general notation ω_i to stress the fact that we are dealing with a scenario.

A fundamental issue is the handling of the information structure, namely what is known at period t when decision x_t, associated with this period, is made. We assume the following structure:

Assumption 2. *In period t, the history of the stochastic process is known up to its current realization ξ_t. This means that the decision-maker is informed of the streams Ξ_t and X_{t-1}. However, the decision-maker has only probabilistic knowledge of the future evolution $\xi_{t+1}, \ldots, \xi_T$, conditionally to the past history Ξ_t.*

The decision process has then the form:

$$\begin{matrix} \text{decision} & \rightarrow & \text{observation} & \rightarrow & \text{decision} & \ldots\rightarrow & \text{decision} & \rightarrow & \text{observation} & \rightarrow & \text{decision} \\ x_0 & & \Xi_1 & & x_1 & & x_{T-1} & & \Xi_T & & x_T \end{matrix}$$

The deterministic equivalent To account for Assumption 2, the variable X_t is formally expressed as the function $X_t(\Xi_t)$. We obtain the following version of the deterministic equivalent

$$
\begin{aligned}
\min \quad & h_0(x_0) + \sum_{\omega \in \Omega} \pi_\omega \left[\sum_{t=1}^{T} h_t(X_t(\Xi_t(\omega)), \Xi_t(\omega)) \right] && \text{(3a)}\\
\text{s.t.} \quad & f_t(X_t(\Xi_t(\omega)), \Xi_t(\omega)) = 0, \qquad t = 1, \ldots, T,\ \omega \in \Omega, && \text{(3b)}\\
& f_0(x_0) = 0, && \text{(3c)}\\
& g_t(X_t(\Xi_t(\omega)), \Xi_t(\omega)) \le 0, \qquad t = 1, \ldots, T,\ \omega \in \Omega, && \text{(3d)}\\
& g_0(x_0) \le 0. && \text{(3e)}
\end{aligned}
$$

However, the explicit formulation of this deterministic equivalent as a function of the decision variables $X_t(\Xi_t(\omega))$ and of the stochastic process $\Xi_t(\omega)$ along the N different scenarios is not a straightforward issue. Problem (3) is not a standard mathematical programming formulation in finite dimension. The transformation into a finite dimension problem can be done via an event tree representation that we discuss in the next section.

3 The event tree formulation

Under Assumption 2, the random data can be represented on an event tree. Without loss of generality, the tree is assumed to be rooted at a single node in period 0. The arcs only link nodes in successive time periods. A node in period t has a unique predecessor in period $t-1$, but possibly several successors in time $t+1$. We can univocally define a node of the event tree as a pair (t, n), $n \in S_t$, where S_t is the set of indices of the nodes at period t. Each node (t, n) on the tree has an unconditional probability $P_{t,n}$ of being visited by the stochastic process. Let us furthermore define N_t as the cardinality of S_t, i.e. $N_t = |S_t|$. Note that $N_t \le N$, for $0 \le t \le T$ and $N_T = N$.

To navigate in the event tree, one has to define a predecessor function that identifies the node that immediately precedes the current node. To this end, we introduce the concept of *predecessor function* $a(t, n, k)$ which maps the current node (t, n) to the index of its predecessor node in period $t-k$, along the unique path that goes from the root $(0, 1)$ to the node (t, n). Using this concept, a scenario i, that ends at node (T, i), is uniquely associated with the sequence of nodes

$$
\Big((0,1), (1, a(T,i,T-1)), \ldots, (T-t, a(T,i,t)), \ldots, (T-1, a(T,i,1)), (T,i)\Big). \tag{4}
$$

Let us consider $\Xi_t(\omega_i)$ and $X_t(\Xi_t(\omega_i))$ the realizations of the stochastic process and of the decision variables along the scenario ω_i. (Recall that we identify ω_i with its index i.) Denoting the unique node that scenario ω_i crosses at time t as $(t, n_i) = (t, a(T, i, T-t))$, with $a(T, i, T-t) \in S_t$, one can introduce the notation

$$\Xi_t(\omega_i) = \Xi_{t,n_i} := (\xi_{1,a(t,n_i,t-1)}, \dots, \xi_{t-1,a(t,n_i,1)}, \xi_{t,n_i}), \tag{5}$$
$$X_t(\Xi_t(\omega_i)) = X_{t,n_i} := (x_{0,1}, \dots, x_{t-1,a(t,n_i,1)}, x_{t,n_i}), \tag{6}$$

with the decision variables x_{t,n_i} and the random process value ξ_{t,n_i} associated with the different nodes of the event tree. We can thus formally replace $\Xi_t(\omega_i)$ and $X_t(\Xi_t(\omega_i))$ by Ξ_{t,n_i} and $X_t(\Xi_{t,n_i})$.

Along these lines, the deterministic equivalent is formulated as the finite dimensional problem

$$\min \; h_0(x_{0,1}) + \sum_{t=1}^{T} \left[\sum_{n \in S_t} P_{t,n}\, h_t(X_{t,n}, \Xi_{t,n}) \right] \tag{7a}$$
$$\text{s.t.} \; f_t(X_{t,n}, \Xi_{t,n}) = 0, \qquad t = 1, \dots, T, \; n \in S_t, \tag{7b}$$
$$f_0(x_{0,1}) = 0, \tag{7c}$$
$$g_t(X_{t,n}, \Xi_{t,n}) \leq 0, \qquad t = 1, \dots, T, \; n \in S_t, \tag{7d}$$
$$g_0(x_{0,1}) \leq 0. \tag{7e}$$

It can thus be seen that the components necessary to define the deterministic equivalent (7) are the following:

1. The time horizon T.
2. The parameters describing the event tree, namely

$$S_t : \text{the set of node indices at period } t, \tag{8}$$
$$a(t,n,k) : \text{the index of the predecessor of node } (t,n) \text{ in period } t-k. \tag{9}$$

3. The stochastic process values and the node probabilities, formally given by

$$\xi_{t,n} : \text{the random process value at node } (t,n), \tag{10}$$
$$P_{t,n} : \text{the unconditional probability of visiting node } (t,n). \tag{11}$$

4. The deterministic parameters.
5. The variables (in general, hooked on nodes).
6. The objective and the constraints (in general, hooked on nodes).

Item 1 is read from the model file for the deterministic problem. Items 2 and 3 are provided by special files. The user is not supposed to edit (or even to look at those files). However, in the case of a transition-based representation (see section 5.2), the stochastic process values are problem dependent. It may be necessary to compute them and the user has to provide the relevant formulas. The last 3 items are modifications and extensions of the deterministic model, according to the information introduced in items 2 and 3.

The key points of the paper are:

- The model information (not the data) in items 2 and 3 is independent of the problem (to the exception of special process value computation in a transition-based representation of the event tree).
- The user can rely on a script to create items 4, 5 and 6 from the deterministic problem and the information in items 2 and 3.

4 An illustrative example

4.1 The deterministic model

To illustrate problem (1), we propose the simple deterministic problem

$$\begin{aligned} \min \quad & (x_2 - \bar{x}_2)^2 && \text{(12a)} \\ & x_t = x_{t-1} + u_{t-1} - \xi_t, \quad t = 1, 2, && \text{(12b)} \\ & 0 \le u_t \le b_t, \quad t = 0, 1, && \text{(12c)} \\ & x_0 = \bar{x}_0. && \text{(12d)} \end{aligned}$$

The decision variables are x_t and u_t, which can be respectively interpreted as state and control variables. The parameters are $\bar{x}_0$, $\bar{x}_2$, b_t and ξ_t. The AMPL formulation and the data file of the deterministic problem are given in Figure 1.

4.2 The stochastic process

As in section 2, we introduce randomness in the problem via the parameter ξ_t. The process has three periods (0, 1, 2). The stochastic process is defined for periods 1 and 2, as period 0 constitutes in fact an auxiliary initial period. There are two transitions per period. Figure 2 displays a classical representation of the process via scenarios on an event tree.

We will show in the next section how this process can be represented in two ways, either by scenarios or by transitions and how a deterministic equivalent can be coupled with both representations.

4.3 The deterministic equivalent

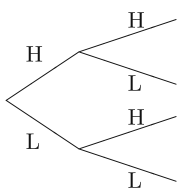
For this example, along the lines of (7) and assuming that the parameters (8)–(11) are known, the deterministic equivalent can be formally written as

$$\begin{aligned} \min \quad & \sum_{n \in S_2} P_{2,n} \, (x_{2,n} - \bar{x}_2)^2 && \text{(13a)} \\ \text{s.t.} \quad & x_{t,n} = x_{t-1,a(t,n,1)} + u_{t-1,a(t,n,1)} - \xi_{t,n}, \quad t = 1, 2, \quad n \in S_t, && \text{(13b)} \\ & 0 \le u_{t,n} \le b_t, \quad t = 0, 1, \quad n \in S_t, && \text{(13c)} \\ & x_{0,1} = \bar{x}_0. && \text{(13d)} \end{aligned}$$

```
#------------------------------------#
# Model for the deterministic problem #
#------------------------------------#

param T;                 # number of periods (time horizon)
set TIME := 0..T;
set TIME1:= 0..T-1;
set TIME2:= 1..T;
set TIME3:= {T};
set TIME4:= {0};
param x0        >= 0;   # initial state
param xTobj     >= 0;   # objective for the state
param b {TIME1} >= 0;   # bound for the control at each period
param xi{TIME2} ;       # perturbation
var x {TIME}    >= 0;   # state of the system at each period
var u {TIME1}   >= 0;   # control for each period
# Objective function
minimize slack: sum{t in TIME3} (x[t] - xTobj)^2;
# System dynamics
subject to timedynamics {t in TIME2}: x[t] = x[t-1] + u[t-1] - xi[t];
# Bound for the control variable
subject to timebound {t in TIME1}: u[t] <= b[t];
# Initial state of the system
subject to initial {t in TIME4} : x[t] = x0;

#-----------------------------------#
# Data for the deterministic problem #
#-----------------------------------#

param T     :=  2;          # number of periods (time horizon)
param x0    :=  0;          # initial state
param xTobj := 10;          # objective for the state
param b     := 0 10 1 10;   # bound for the control at each period
param xi    := 1 8  2 12;   # perturbation
```

Fig. 1 Model and data for the deterministic problem in AMPL

Fig. 2 Stochastic process in the illustrative example



Scenario	ξ_1	ξ_2	π
HH	8	12	0.36
HL	8	10	0.24
LH	6	10	0.24
LL	6	8	0.16

The AMPL formulation[2] of this deterministic equivalent is displayed in Figure 3. It clearly appears now that the main difficulty is the automatic computation of parameters (8)–(11) from the deterministic model and the stochastic process description. This issue is addressed in the next sections.

[2] The functions `nodeprob[·,·]`, `NodeSet[·]` and `kancest[·,·,·]` that appear in that formulation will be defined in the next sections.

```
#-------------------------------------------------#
# Generated model for the deterministic equivalent #
#-------------------------------------------------#
# PARAMETERS.
param  T;
set TIME := 0..T;
set TIME1:= 0..T-1;
set TIME2:= 1..T;
set TIME3:= {T};
set TIME4:= {0};
#
# PROBABILISTIC SECTION OF THE MODEL
# insert here the formulas to compute the functions relative
# to the event tree (Set of nodes, k-ancestor function, etc.)
# See next sections.
#
param  x0        >= 0;
param  xTobj     >= 0;
param  b {TIME1} >= 0;
param  xi{t in TIME2,n in NodeSet[t]};

# VARIABLES.
var    x {t in TIME, n in NodeSet[t]}>= 0;
var    u {t in TIME1,n in NodeSet[t]}>= 0;

# OBJECTIVE FUNCTION.
minimize slack: sum{t in TIME3,n in NodeSet[t]} nodeprob[t,n]*((x[t,n] - xTobj)^2);

# CONSTRAINTS.
subject to  timedynamics {t in TIME2, n in NodeSet[t]}:
       x[t,n] = x[t-1,kancest[t,n,1]] + u[t-1,kancest[t,n,1]] - xi[t,n];
subject to  timebound {t in TIME1, n in NodeSet[t]}   : u[t,n] <= b[t];
subject to  initial {t in TIME4, n in NodeSet[t]}     : x[t,n] = x0;
```

Fig. 3 The deterministic equivalent problem in AMPL

5 Primitives on the event tree

We discuss two ways to describe an event tree and we give the corresponding parameters and auxiliary functions. In the first case, the base concept is a scenario. The scenarios are described recursively, starting from an initial scenario. A new scenario is linked to an existing one, which becomes its parent scenario. The scenario description includes the values taken by the stochastic process. In the second approach, the base concept is the transition process. Starting from the root node, the set of possible transitions recursively defines the nodes in the future periods. The index of a node in period *t* is thus associated with the state of the process. Contrary to the scenario description, the event tree and the values taken by the stochastic process are not given explicitly; they have to be built from base information.

The scenarios provide an explicit description of the event tree and, in particular, the values taken by the stochastic process along the scenarios are explicitly given.

The transition-based approach is well-fitted for a discrete-time stochastic process with a known finite one-step transition distribution. This approach can be furthermore exploited to implement complex discretization procedure of underlying random variables (see for example van Delft and Vial 2004). The scenario-based formulation is usually the output of a complex procedure aiming to approximate a continuous time and state process by a discrete event tree through Monte-Carlo simulation and scenario reduction schemes whose description is out of the scope of this paper. See for instance Høyland and Wallace (2001) or Pflug (1989).

The two approaches will now be described in detail. From here on, we choose to insert the argument of a function within square brackets '[' and ']'. This notation matches that which is in force in the algebraic modeling language AMPL, which we use to model and solve the examples (see van Delft and Vial 2004).

5.1 Scenario-based representation

5.1.1 The general setting

As previously defined, a scenario is a realization of the underlying stochastic process, from $t = 0$ to the horizon $t = T$. The first scenario in the list is often named the *base scenario*: all other scenarios will be recursively defined with respect to this base scenario.

In fact, it is useful to characterize how scenarios are related to each other. First, from the definitions, it is directly seen that two scenarios $\omega[i]$ and $\omega[j]$ may coincide up to a certain time t', but if these scenarios are distinct at t', they are disjoint for all periods $t > t'$. In contrast, if two scenarios are identical in period t, they are indistinguishable for all periods $t' < t$. This can be interpreted by considering that $\omega[i]$ is the *parent* scenario for scenario $\omega[j]$ and the meeting period is t'. Formally, we define two parameters $\sigma[i]$ and $\tau[i]$, with the understanding that $\sigma[i]$ is the index of the parent scenario for scenario $\omega[i]$, and $\tau[i]$ is the meeting period with its parent scenario. It is worth noting that under this interpretation, the set $S[t]$ can be viewed as the set of distinct scenarios at time t.

We recall that the user gives the probabilities $\pi[n]$ associated with each scenario $\omega[n] = n$. The values taken by the stochastic process along scenario $\omega[n]$ are denoted by $\Xi[t, \omega[n]]$ and are entered as data, or computed by generic formulas. The set of node indices at each period is then computed as

$$S[t] = \begin{cases} \{1\} & \text{if } t = 0, \\ \{n \mid \omega[n] \in \Omega, \tau[n] < t\} & \text{if } 1 \leq t \leq T. \end{cases} \tag{14}$$

Under this interpretation, for each time period $t \leq T$, the set $S[t]$ is viewed as the set of scenarios which have diverged at some time $t' < t$. This scenario description is illustrated via the example, in Figure 4 below.

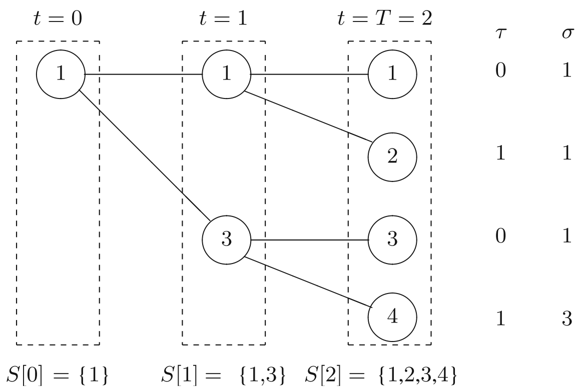


Fig. 4 Scenario description for the illustrative example

Furthermore, for each node (t,n), the unconditional probability of being visited by the stochastic process can be recursively computed as follows from the scenario probabilities

$$P[t,n] = \begin{cases} \pi[n] & \text{if } t = T, \\ P[t+1,n] + \sum\limits_{\substack{m \in S[t+1] \\ \tau[m]=t,\, \sigma[m]=n}} P[t+1,m] & \text{if } 0 \le t < T. \end{cases} \tag{15}$$

The one-period predecessor function $a[t,n,1]$ is defined by

$$a[t,n,1] = \begin{cases} n & \text{if } t > \tau[n]+1, \\ \sigma[n] & \text{if } 1 \le t \le \tau[n]+1, \end{cases} \tag{16}$$

with $a[t,1,1] = 1$, for $1 \le t \le T$.

The k-period predecessor function $a[t,n,k]$ can afterwards be defined as

$$a[t,n,k] = \begin{cases} a[t,n,1] & \text{if } k = 1, \\ a[t-1, a[t,n,1], k-1] & \text{otherwise,} \end{cases} \tag{17}$$

for all $1 \le k \le t$.

5.1.2 Information provided by the user

The set of scenarios[3]	:	$1, \ldots, N$	
Scenario probabilities	:	$\pi[n]$	for $1 \le n \le N$
Parent scenario	:	$\sigma[n]$	for $1 \le n \le N$
Meeting period	:	$\tau[n]$	for $1 \le n \le N$
Process values	:	$\xi[\tau[n]+1,n], \ldots, \xi[T,n]$	for $1 \le n \le N$.

[3] In the present AMPL file, the scenarios can be alpha-numerically named, implying a few simple changes, the most important being $S[0] = \{\text{root}\}$ instead of $\{1\}$.

5.1.3 Generic information

Set of nodes	:	$S[t]$	computed by formula (14)
Node probabilities	:	$P[t,n]$	computed by formula (15)
Predecessor function	:	$a[t,n,1]$	computed by formula (16)
k-predecessor function	:	$a[t,n,k]$	computed by formula (17).

5.1.4 Notation in the AMPL models

We have chosen the following AMPL notations

The set of scenarios	:	$1,\dots,N$	$\rightarrow$	`Scen`
Scenario probabilities	:	π	$\rightarrow$	`scenprob`
Parent scenario	:	σ	$\rightarrow$	`parent`
Meeting period	:	τ	$\rightarrow$	`meet`
Process values	:	ξ	$\rightarrow$	`xi`
Set of nodes	:	S	$\rightarrow$	`NodeSet`
Node probabilities	:	P	$\rightarrow$	`nodeprob`
k-predecessor function	:	a	$\rightarrow$	`kancest`.

The corresponding AMPL set of instructions is displayed in Figure 5.

With this scenario-based description, the deterministic equivalent is easily obtained from the deterministic model by making the variable x and the parameter ξ contingent on the nodes (t,n) of the tree represented in Figure 2. The AMPL formulation of the deterministic equivalent has been given in Figure 3.

One notices that the k-ancestor function `kancest`$[\cdot,\cdot,\cdot]$ and the set of indices `NodeSet`$[\cdot]$, that are used in the model are not defined in Figure 2 (see footnote 2, page 25). These functions are defined via a set of generic AMPL instructions that will be appended to the deterministic file by an automatized process to be described in the next section. Furthermore, in the proposed AMPL formulation without loss of generality, we formally define the different sets of time periods involved in the problem definition. This is a convention that will be explained in section 6.

5.2 Transition-based representation

5.2.1 The general setting

In the transition-based representation, the user describes the stochastic process by a one-step transition process from one period to the next fully explained in this section. This information is used to build the tree and to compute the probabilities and the stochastic process values to be assigned at each node. It is usually the case that the process value at a given node in $t+1$ depends on two elements. First, it depends on the history of the stochastic process until t, namely $\Xi[t]$. Second, it also depends on the nature of the transition from t to $t+1$. We

```
#----------------------------------------------------------------#
# Computation of the primitives: the scenario-based representation #
#                       The formulas                               #
#----------------------------------------------------------------#
#
# DATA STATEMENT
set   Scen;
param scenprob {Scen};
param parent   {Scen} symbolic in Scen union {'root'};
param meet     {Scen};
#
# GENERIC FORMULAS
set   NodeSet {t in 0..T}:= if t = 0 then {'root'} else {s in Scen: meet[s]<t};
param ancest  {t in 1..T, n in NodeSet[t]} symbolic in Scen union {'root'}:=
 if t = 1 then 'root'
          else if t > meet[n] + 1 then n
                                 else parent[n];
param kancest {t in 1..T, n in NodeSet[t], k in 1..t} symbolic in Scen union {'root'}:=
 if t = 1 or k = 1 then ancest[t,n]
                  else kancest[t-1,ancest[t,n],k-1];
set   Reverse := T..0 by -1;
param nodeprob {t in Reverse, n in NodeSet[t]}:=
 if t = T then scenprob[n] else if t = 0 then 1
 else nodeprob[t+1,n] + sum{m in NodeSet[t+1]: meet[m]=t and parent[m]=n} nodeprob[t+1,m];

#----------------------------------------------------------------#
# Computation of the primitives: the scenario-based representation #
#                          The data                                #
#----------------------------------------------------------------#
#
# data concerning the scenarios
set    Scen := hh hl ll lh;
param  :meet    parent  scenprob :=
hh        0      root    .36
hl        1      hh      .24
ll        0      root    .16
lh        1      ll      .24 ;
#
# data concerning the stochastic process
param  xi : hh  hl  ll  lh :=
1            8   .   6   .
2           12  10   8  10  ;
```

Fig. 5 Event tree parameters for a scenario-based representation

thus assume that the description of the one-step transition process from a node in period t to a node in period $t+1$ is made via the following functions

- $f[t]$:= the number of transitions from any node at period t[4],
- $p[t,j]$:= the one-step transition probability of transition $j \in \{1,\ldots,f[t]\}$,
- $\epsilon[t,j]$:= the one-step random factor corresponding to transition $j \in \{1,\ldots,f[t]\}$.

The event tree is then built as follows. At period t, the nodes are numbered from 1 to $N[t]$ going from top to bottom: node (t,n) is the n-th node from the

4 In this paper, we restrict our presentation to the symmetric case where the number of transitions depends only on the time period. However, models with asymmetric event trees can be handled. They involve computed parameters $f[t,n]$, contingent on the information (probability, value of the stochastic process) available at node (t,n). This case has been treated in van Delft and Vial (2004).

top in period t. In this way, we directly find $S[t] = \{1, ..., N[t]\}$. The number of nodes at period t is recursively computed by

$$N[t] = N[t-1]\ f[t-1], \tag{18}$$

with $N[0] = 1$, where $f[t]$ is the number of branches emanating from each node at period t. In this formulation, the one-period predecessor function $a[t, n, 1]$ is recursively defined by

$$a[t,n,1] = \begin{cases} a[t,n-1,1] & \text{if } n \le a[t,n-1,1]f[t-1], \\ a[t,n-1,1]+1 & \text{otherwise,} \end{cases} \tag{19}$$

with

$$\begin{aligned} a[t,1,1] &= 1, \quad \text{for } 0 < t \le T, \\ a[t,n,0] &= n, \quad \text{for } 0 \le t \le T, \qquad n \in S[t]. \end{aligned}$$

Then we recursively define the general k-period predecessor function with (17).

We also need to identify the transition index j that led from the previous node $(t-1, a[t, n, 1])$ to the actual node (t, n). It can be formally computed by the auxiliary function $\ell[t, n]$ defined as

$$\ell[t,n] = n - (a[t,n,1]-1)f[t-1]. \tag{20}$$

Using the primitives (19) and (20) one may write

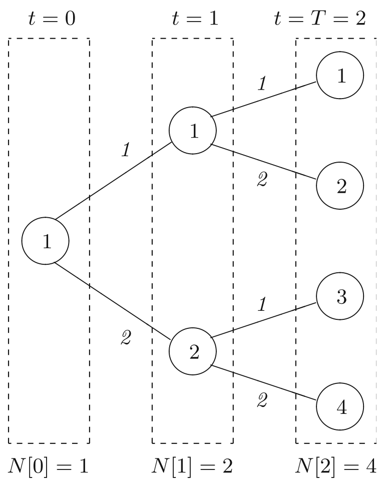
$$\xi[t,n] = q[\Xi[t-1, a[t,n,1]], \epsilon[t-1, \ell[t,n]]], \tag{21}$$

where $q[\cdot,\cdot]$ is a given function. According to (21), it can be seen that the random process value $\xi[t, n]$ is computed based on past events $\Xi[t-1, a[t, n, 1]$ and on the one step transition process $\epsilon[t-1, \ell[t, n]]$, corresponding to the transition leading to node (t, n).

To illustrate our point, we consider the illustrative example, which is a three-period model (with $T = 2$) with a symmetric event tree. We represent it in Figure 6. The uncertainty unfolds in time from left to right. Nodes appearing in the same vertical cut belong to the same time period. At $t = 0$, there are two branches giving rise to a pair of nodes in $t = 1$ ($f[0] = 2$ and $N[1] = 2$). At $t = 1$, each node has two branches ($f[1] = 2$) and there are $N[2] = 4$ nodes in $t = 2$. Time $t = 2$ is the horizon: no branch emanates from those nodes. Furthermore, the transition indices are the following : $\ell[1, 1] = 1$, $\ell[1, 2] = 2$, $\ell[2, 1] = 1$, $\ell[2, 2] = 2$, $\ell[2, 3] = 1$ and $\ell[2, 4] = 2$.

From our assumptions, for a given node (t, n), the conditional transition probabilities depend on the period t and on the transition index j. In the present formulation, those probabilities are given and denoted $p[t, j]$. Often, these probabilities result from the discretization procedure of an underlying random variable (see, for example, van Delft and Vial (2004)). From these transition

Fig. 6 Transition-based representation for the example



probabilities, the unconditional occurrence probability $P[t,n]$ of node (t,n) can be recursively computed by

$$P[t,n] = \begin{cases} 1 & \text{if } t = 0, \\ p[t-1,\ell[t,n]]\,P[t-1,a[t,n,1]] & \text{if } 1 \leq t \leq T. \end{cases} \qquad (22)$$

Figure 7 displays the formulas written in AMPL.

5.2.2 *Information provided by the user*

Number of emanating arcs	:	$f[t]$,	for $0 \leq t < T$,
One-step random factor	:	$\epsilon[t,j]$,	for $0 \leq t < T$, $j = 1,\ldots,f[t]$,
Transition probabilities	:	$p[t,j]$,	for $0 \leq t < T$, $j = 1,\ldots,f[t]$.

5.2.3 *Generic information*

Number of nodes	:	$N[t]$	computed by formula (18).
Set of nodes	:	$S[t] = \{1,\ldots,N[t]\}$	
Predecessor	:	$a[t,n,1]$	computed by formula (19).
k-predecessor	:	$a[t,n,k]$	computed by formula (17).
Transition index	:	$\ell[t,n]$	computed by formula (20).
Node probabilities	:	$P[t,n]$	computed by formula (22).

```
#-------------------------------------------------------------------#
# Computation of the primitives: the transition-based representation #
#                            The formulas                             #
#-------------------------------------------------------------------#
#
# declaration of data given by the user :
param transnb   {t in 0..T-1};
param transprob {t in 0..T-1, k in 1..transnb[t]};

# automatically computed values
param nodenb    {t in 0..T} := if t = 0 then 1 else nodenb[t-1]*transnb[t-1];
set   NodeSet   {t in 0..T} := {1..nodenb[t]};
param ancest    {t in 1..T, n in NodeSet[t]}:= if t = 1 then 1 else if n = 1 then 1
  else if n <= ancest[t,n-1]*transnb[t-1] then ancest[t,n-1] else ancest[t,n-1]+1;
param kancest   {t in 1..T, n in NodeSet[t], k in 1..t}:= if t = 1 or k = 1
  then ancest[t,n] else kancest[t-1,ancest[t,n],k-1];
param lastmove  {t in 1..T, n in NodeSet[t]}:=  n-(ancest[t,n]-1)*transnb[t-1];
param nodeprob  {t in 0..T, n in NodeSet[t]}:= if t = 0 then 1
  else transprob[t-1,lastmove[t,n]]*nodeprob[t-1,ancest[t,n]];

#-------------------------------------------------------------------#
# Computation of the primitives: the transition-based representation #
#                            The data#
#-------------------------------------------------------------------#
#
# data concerning the transitions
param transnb   :=0 2   1 2;
param transprob :1  2   :=
0       0.6 0.4
1       0.6 0.4 ;

# data concerning the stochastic process
param epsilon : 0   1   :=
0       8   6
1       4   2  ;
```

Fig. 7 Event tree parameters for a transition-based representation

5.2.4 *Notation in the AMPL models*

In complement to the AMPL notations introduced in the scenario-based approach, we add the following notations

Number of nodes	:	N	$\rightarrow$	`nodenb`
Number of transitions	:	f	$\rightarrow$	`transnb`
Transition probabilities	:	p	$\rightarrow$	`transprob`
One-step random factor	:	ϵ	$\rightarrow$	`epsilon`
Last transition index	:	ℓ	$\rightarrow$	`lastmove`.

5.2.5 *Computing the values of the stochastic process*

We have explained in the general setting that in the transition-based representation, the stochastic process values to be assigned at each node have to be computed. It can be seen via (21) that these values are assumed to depend on the history of the stochastic process and on the nature of the last transition. If this dependence is described by a formula, this formula must be made contingent on the node of the tree, which is not direct for a user, as the tree is not externally given (as in the scenario-based approach). To permit an automatic computation, we ask the user to modify the deterministic model (1) by adding the following deterministic equation

$$\xi[t] = q(\Xi[t-1], \epsilon[t-1]), \tag{23}$$

that expresses $\xi[t+1]$ as a function of the external parameter $\epsilon[t]$. In the AMPL deterministic model, $\xi[t]$ will be declared as a variable,[5] $\epsilon[t]$ as a parameter and the transition equation (23) as a constraint. The parameter $\epsilon[t]$ can be viewed as a pure random effect that is node independent. With such conventions, the contingent process values will be automatically computed during the deterministic equivalent generation.

We illustrate this approach with our example. In this example, the stochastic process in Figure 2 can also be alternatively represented by the transition function

$$\xi[t] = \xi[t-1] + \epsilon[t-1], \tag{24}$$

with the understanding that $\xi[0] = 0$ is fixed and $\epsilon[t]$ is the underlying random effect. Table 1 displays these data.

Table 1 Data on the transition process

	Outcomes			
	$t = 0$		$t = 1$	
$\epsilon[t]$	8	6	4	2
$p[t]$	0.6	0.4	0.6	0.4

We thus reformulate the deterministic problem (12) for the example as

$$\min \quad (x[2] - \bar{x}[2])^2 \tag{25a}$$

$$x[t] = x[t-1] + u[t-1] - \xi[t], \quad t = 1, 2, \tag{25b}$$

$$0 \le u[t] \le b[t], \quad t = 0, 1, \tag{25c}$$

$$\xi[t] = \xi[t-1] + \epsilon[t-1], \quad t = 1, 2, \tag{25d}$$

$$x[0] = \bar{x}[0],\ \xi[0] = 0. \tag{25e}$$

In this formulation, the parameter ξ is treated as a variable, but ϵ is the random parameter that triggers the process. The change in the status of ξ is purely formal, but it gives a common framework for the description of process via

[5] These variables will take fixed values on the event tree, a fact that the solver preprocess will use to eliminate them for the solution phase.

scenario or via transitions. The modification in the deterministic model (Figure 1) is displayed in Figure 8.

```
#-----------------------------------------------------------------#
# Computation of the values of the stochastic process             #
#           Extension of the deterministic model                  #
#-----------------------------------------------------------------#
#
param  xi0 := 0;        # initial value for the stochastic process
var    xi {TIME};       # stochastic process
param  epsilon{TIME1};  # underlying randomness
#
subject to initialstate    {t in TIME4}: xi[t] = xi0;
subject to processdynamics {t in TIME2}: xi[t] = xi[t-1] + epsilon[t-1];
```

Fig. 8 AMPL formulation of the dynamics of the perturbation

6 Automatic generation of the deterministic equivalent in AMPL

In this section, we discuss the automatic merging of the deterministic structure of the optimization problem ($f[t]$, $f[0]$, $g[t]$ and $g[0]$) and the stochastic information embedded in the event tree. We recall here that, while it is relatively easy to describe the two base components – the underlying deterministic model and the stochastic process – it is tedious to define the contingent variables and constraints and build the deterministic equivalent. We show in this section how to define a systematic procedure that performs this. This procedure can then be translated into a script, so that the user is completely relieved from the painstaking work. The procedure is simpler in the case of a scenario-based description of the stochastic process. We shall detail this case and briefly review the few modifications required to handle the transition-based case.

6.1 The two basic components

The first step in the generation of the deterministic equivalent is the description of the two basic components underlying the stochastic model, namely the deterministic structure and the full description of the stochastic process via an event tree. This step, which is to be performed by the user, can be summarized as follows:

1. Formulate the underlying deterministic dynamic model as defined in (1).
   – The number of time periods must be defined as a parameter, named "T".
2. Enter the information describing the structure and the values of the stochastic process as described in section 5.
   – To do this, the convention described in subsection 5.1.4 should be followed.

– The set of scenarios $S[T]$ must be non-empty, and the first scenario in S[T] must have no parent scenario. The user must describe at least one scenario, named scenario 1.

6.2 Building the deterministic equivalent

The second step consists in building the deterministic equivalent. This can be performed in an automatic fashion provided the deterministic model (1) is written according to some simple conventions. We detail first the conventions, and describe later the transformation rules that map the deterministic dynamic problem to the deterministic equivalent.

6.2.1 Notation for the deterministic model

The formulation must comply with conventions. The first convention is the standard modeling convention in AMPL.

Convention 1. *Indices are entered as arguments of functions, i.e., they are delimited by square brackets "*[*" and "*]*". For instance, we write*

$$f_t(x_0, x_1, \ldots, x_t, \xi_1, \xi_2, \ldots, \xi_t) \quad \textit{as} \quad f[t, x[0], x[1], \ldots, x[t], \xi[1], \xi[2], \ldots, \xi[t]].$$

The next convention applies to variables and parameters whose values are contingent on the nodes of the event tree.

Convention 2. *The time index at which the value of the variable or the parameter is fixed is first in the list of indices which influence the value of the variable or parameter.*

For instance, we write $x[t, j]$, with j representing a location or a product type, and not $x[j, t]$.

In the declaration phase, one must define the set of all time periods, and possibly some of its subsets.

Convention 3. *Any set of time periods starts with the same character string.*

Our own convention is to denote `TIME` as the full set of time periods $\{0, \ldots, T\}$. The denomination of any subset of `TIME` will take the form `TIME<exp>`, e.g., `TIMEa` $= \{2, \ldots, T-1\}$.

In the problem definition (objective and constraints), the following convention is in force:

Convention 4. *Time indices in variables and parameters are never replaced by constants.*

For instance, we do not write $x[0] = 0$, but, in mathematical style, $\{x[t] = 0 \mid t = 0\}$.

Under these conventions, the mathematical programming formulation can be obtained from (1) by some simple transformations described below.

6.2.2 Transformation rules

The main task in defining the deterministic equivalent consists in properly associating variables and constraints to the different nodes of the event tree. Furthermore, the objective function has to be reformulated along (3). This can be done via the following rules.

Rule 1 (Variable transformation). *Replace the variable X_t with the $N[t]$ variables $X[t,n]$, $n \in S[t]$. If a variable has two time indices, only the first time index, corresponding to the period when the decision is made, is to be taken into account.*

Rule 2 (Parameter transformation). *Replace the parameter Ξ_t with the $N[t]$ parameters $\Xi[t,n]$ where $n \in S[t]$.*

Rule 3 (Constraint transformation). *Replace the constraints*

$$f_t(X_t, \Xi_t) = 0 \text{ and } g_t(X_t, \Xi_t) \leq 0$$

with the $N[t]$ constraints

$$f[t, X[t,n], \Xi[t,n]] = 0 \text{ and } g[t, X[t,n], \Xi[t,n]] \leq 0,$$

where $n \in S[t]$.

Rule 4 (Objective transformation). *In the objective, replace*

$$h_t(X_t, \Xi_t) \quad \text{with} \quad \sum_{n \in S[t]} P[t,n]\, h[t, X[t,n], \Xi[t,n]].$$

6.2.3 Implementing the rules

We propose on the internet[6] a script that implements the rules via the syntactic interpretation language PERL (Wall et al. 2001). This script is generic and applies to any model and data that comply with our rules.[7] The script automatically generates the corresponding deterministic equivalent. It performs a syntactic analysis based on regular expressions (see Friedl 1997).

Inputs–outputs
The script counts four input files and two output files. We describe first the input:

[6] The "DET2STO" program. Free program available online at the website: "logilab.unige.ch".

[7] The authors cannot guarantee that all syntaxes supported by AMPL are currently understood by the script.

File I.1 describes the deterministic dynamic model structure (1) (.mod format in AMPL).
File I.2 describes the corresponding numerical data (.dat).
File I.3 describes the stochastic primitives (.mod).
File I.4 describes the numerical data underlying the stochastic process (.dat).

And then the output:

File O.1 describes the deterministic equivalent model structure (7) (.mod).
File O.2 describes the corresponding numerical data (.dat).

Script structure
First, the script generates the deterministic equivalent model, as follows:

- Pre-process in order to check the input structure validity.
- Declare and define the primitives according to the formulas in subsections 5.2.3 and 5.1.3.
- Identify declarations of variables and parameters.
- Modify the above declarations according to rule 1.
- Identify definitions of constraints.
- Modify the above definitions according to rule 2.
- Identify the definition of the objective function.
- Modify the above definition according to rule 3.

In a second phase, the script generates the deterministic equivalent data file by merging the two numerical data input files.

The rules are simple enough to encode, but, in practice, the user must take great care to make sure that the input files are consistent and complete. One must keep in mind that there are often many different ways to write a model in AMPL, only some of which are consistent with Conventions 1–4.

Recall that a generated deterministic equivalent model is presented in Figure 3.

6.3 The case of a transition-based formulation

The reader has certainly noticed that the transition-based formulation is more complicated than the scenario-based one. Indeed, the user has to provide information on the dynamics of the stochastic process as formulas that can be handled in AMPL. The surprising fact is that the deterministic equivalent can still be produced by a script file at a minimal additional effort from the user.

The transformation rules differ from the scenario-based approach. Rule 1 is replaced by:

Rule 5 (Variable transformation). *Replace the variables X_t and Ξ_t with the $N[t]$ variables $X[t,n]$ and $\Xi[t,n]$, $n \in S[t]$.*

Rule 2 is replaced by the following:

Rule 6 (Parameter transformation). *Replace the parameter ϵ_{t-1} with the $N[t]$ parameters $\epsilon[t-1, \ell[t,n]]$ where $n \in S[t]$.*

Rules 3 and 4 are maintained as in subsection 6.2.2.

6.4 Extension to variables with time lags

For the sake of simpler notation, we have not considered problems with time-lagged decisions. However it is often the case that a decision is to be taken at t but takes effect at $t+k$. This can be formalized by means of a double-indexing. We adopt the following convention

Convention 5. *If a variable or a parameter includes more than one time index, the first time index corresponds to the time at which the decision is taken; the subsequent time indices define the dates at which the parameter or variable has an impact.*

For instance, $x[t, t+1]$ will be made contingent on nodes $[t, n], n \in S[t]$, but not to nodes $[t+1, m], m \in S[t+1]$.

7 Conclusion

Multistage stochastic programming with recourse is one possible approach to model sequential decision-making under uncertainty. It applies to discrete, or discretized, stochastic processes and leads to the formulation of the *deterministic equivalent*. This mathematical programming problem is expressed in algebraic terms and can be solved by available optimization software. In this paper, we have proposed a methodology to help users to formulate the deterministic equivalent program from their base deterministic model and from general event tree representations of the stochastic process. In particular, we have introduced recursively-defined generic functions that make it possible to navigate through the tree and the alternative event tree formulations.

Together with the methodology, we offer a tool, the PERL script, that fully automatizes the process of of building and solving stochastic programming problems. We would also like to emphasize a nice feature of the proposed tool. Associated with the free GNU linear programming kit that contains the GNU MATHPROG language – a subset of AMPL – and a LP solver, our tool offers a free alternative to stochastic linear programming. It is our hope that the tool will evolve and benefit from contributions of the community. A debugger that would check the correctness of the deterministic model would be most welcome, as well as a user-friendly interface.

Acknowledgements The authors wish to thank anonymous referees for their comments and suggestions that have helped in improving this paper. The first and third authors also acknowledge the support of Fonds National Suisse de la Recherche Scientifique, grant # 100012-105309

References

Birge JR, Louveaux F (1997) Introduction to stochastic programming. Springer, Berlin Heidelberg New York

Condevaux-Lanloy C (2004) Extensions de l'interface entre langages de modélisation et codes d'optimisation: application à la programmation stochastique multi-étapes linéaire et non-linéaire. PhD Thesis, University of Geneva

Condevaux-Lanloy C, Fragnière E, King A (2002) SISP, simplified interface for stochastic programming. Optim Methods Softw 17(3):423–443

Dormer A, Vazacopoulos A, Verma N, Tipi H (2005) Modeling & solving stochastic programming problems in supply chain management using XPRESS-SP. Supply Chain Optimization; Geunes and Pardalos Eds, (2005) 1–27

Edwards J, Birge J, King A, 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. In: Working Paper WP-85-03, International Institute for Applied Systems Analysis, Laxenburg, Austria

Fourer R, Gay D, Kernigham B (1993) AMPL: a modeling language for mathematical programming. The Scientific Press Series, San Francisco

Fourer R, Gay D (1997) Proposals for stochastic programming in the AMPL modeling language. In: Session WE4-G-IN11, International symposium on mathematical programming, lausanne, August 27

Fourer R, Lopes L (2004) A filtration-oriented system for modeling in stochastic programming. In: Technical report, Department of systems and industrial engineering, University of Arizona

Fragnière E, Gondzio J, Sarkissian R, Vial J.-Ph (2000a) Structure exploiting tool in algebraic modeling languages. Manage Sci 46:1145–1158

Fragnière E, Gondzio J, Vial J.-Ph (2000) Building and solving large-scale stochastic programs on an affordable distributed computing system. Ann Oper Res 99(1/4):167–187

Friedl JEF (1997) Mastering regular expressions, O'Reilly & Associates, Cambridge

Gassmann HI, Ireland AM (1996) On the formulation of stochastic linear programs using algebraic modeling languages. Ann Oper Res 64:83–112

Gassmann HI, Schweitzer E (2001) A comprehensive input format for stochastic linear programs. Ann Oper Res 104:89–125

Høyland K, Wallace S (2001) Generating scenario trees for multi-stage decision problems. Manage Sci 47:295–307

IBM (1998) OSL Stochastic Extensions: Guide and Reference, ©IBM Corp.,

Kall P, Wallace S (1994) Stochastic programming. Wiley, New York

Makhorin A (2005) GNU linear programming kit: reference manual. Free software foundation, 4.8

Murtagh B (1981) Advanced linear programming: computation and practice, McGraw-Hill, New York

Pflug G (1989) Optimal scenario tree generation for multiperiod financial planning. Math Program 89:251–271

Valente P, Mitra G, Poojari C, Kyriakis T (2001) Software tools for stochastic programming: a stochastic programming integrated environment (SPInE). In: Technical report, Brunel University

van Delft Ch, Vial J.-Ph (2004) A practical implementation of stochastic programming: an application to the evaluation of option contracts in supply chain. Automatica 40:743–756

Wall L, Christiansen T, Orwant J (2001) Programming PERL, 3rd ed. O'Reilly & Associates, Cambridge