AN XML-BASED SCHEMA FOR STOCHASTIC PROGRAMS

R. Fourer*, H.I. Gassmann†, J. Ma‡ and R.K. Martin§

Revision of November 15, 2006

Abstract

This paper describes a proposed format to record instances of stochastic programs. It forms part of a larger XML-based schema that is designed to allow the expression of essentially any type of mathematical program within a unifying framework. A wide variety of different linear and nonlinear stochastic programs can be handled, and the paper describes in detail how this is done. Screen captures and sample problems illustrate the use of the schema.

1 Introduction

Standard problem descriptions are necessary whenever there is a possibility that a given problem instance is passed to more than one solver. They also allow benchmarking of algorithms, archiving of test problems, and sharing of problems. Particularly the benchmarking of algorithms relies heavily on being able to test a common set of problems on a variety of different programs or solvers.

It seems quite likely that the development of linear programming, especially the emergence of interior point algorithms and the subsequent resurgence of development in the simplex algorithm (notably the dual steepest edge algorithm) would not have happened in the same way it did, had not David Gay made available a set of test problems following the 1985 Mathematical Programming Symposium [16, 27]. What contributed decisively to the success of this netlib collection was an agreed-upon *de facto* standard for the representation of linear programs, the MPS format (see, e.g., [18, 20]).

*Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, Illinois

†School of Business Administration, Dalhousie University, Halifax, Canada, e-mail: hgassman@mgmt.dal.ca (corresponding author)

‡Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, Illinois

§Graduate School of Business, The University of Chicago, Chicago, Illinois

This should be contrasted with the situation in network optimization, where Klingman *et al.* lamented that “this nonstandardization of problem specification [. . .] is most frustrating and has hampered benchmarking” [20]. Clearly, in their view, a commonly accepted way to describe instances of network problems would have been highly desirable.

Stochastic programming has had a common problem description, the SMPS format, since around 1984 [1, 8]. The format has been updated several times over the years [12, 15]. Although it has been adopted by many software developers, the SMPS format does not share the same universal acceptance enjoyed by the MPS format upon which it is based.

Nonlinear programming does not have a common format to represent instances. A file format based on the MPS record structure was put forward by Conn *et al.* [5], but it did not gain wide-spread acceptance. There is also a competing proposal called xMPS [17, 35].

Two general ways of solver communication are practised today: Users can communicate their problems by supplying subroutines that compute function values and occasionally gradients as well as higher-order derivatives. Solvers employ a variety of calling sequences; some use Fortran, others C or C++, and the user is expected to tailor the code to the needs of the solver every time (see, e.g., [18, 25, 33, 34]).

The other general approach consists in defining the problem inside a modelling environment such as AMPL [9] or GAMS [3] and to connect this modelling environment directly into the solver [28, 37, 39]. One disadvantage of this approach lies in the fact that it ties the problem instance to a particular piece of software; anyone intending to solve the problem must have access to this program. Another disadvantage is illustrated in the left panel of figure 1: If there are n different modelling environments and m solvers, $n*m$ different interfaces will have to be developed. An intermediate format such as MPS greatly simplifies the approach, as only $n+m$ interfaces will have to be written, n of them to capture the output from the modelling environments, and m others to read the resulting file into a solver (see figure 1, right panel).

It is true that communication by file is slower than communication in memory. However, in modern systems the solution process often takes place in a distributed environment: one computer prepares the model (often with input from distributed databases), while another computer (or group of computers) is responsible for the solution. This creates considerable traffic over a network, and whether the communications takes place using standardized files or some other format is largely immaterial.

The MPS format, as the SMPS format based upon it, uses very old technology. (For a more detailed explanation of the MPS format, see [24].) It was developed forty years ago for IBM’s Mathematical Programming System and has subsequently been adopted by all commercial LP solvers. The record format is very strict, and the name space is limited to eight characters for each row and column. In addition, only

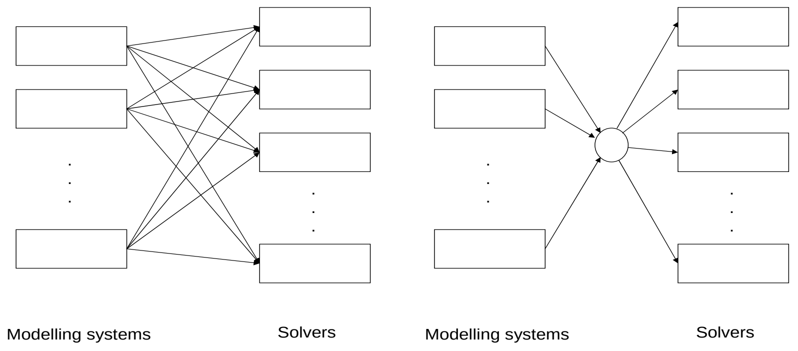

Fig. 1: Interfaces between modelling system and solver without common problem description (left) and with (right).

twelve characters are available to represent the constraint and objective coefficients — including signs and exponents. Hence there is limited accuracy in the input data. Furthermore, there is no way to record whether a problem is intended to be maximized or minimized. Finally, the original format was intended to deal with linear problems only, although in the interim extensions have been added to allow the solution of (mixed) integer linear problems and optimization of quadratic objective functions subject to linear constraints. However, these extensions to the MPS format are not universally agreed upon, and several "flavours" of the format are now in use (see, e.g., [6, 21, 24]). Ironically it must be added that of late many algebraic modelling systems have abandoned the MPS format and reverted to proprietary data formats in order to communicate with the solver directly in memory.

A unifying format to express a wide variety of mathematical programming problems would provide a major service to the optimization community. The OSxL initiative by Optimization Services [22, 23] constitutes just such an effort. It is written in XML (eXtensible Markup Language). Similar to other markup languages such as HTML, XML uses tags to separate data items from one another. Tags are nested in a tree structure, but the definition, position and order of tags is left to the user and can be described in a schema file (which is itself an XML document). (For more information on XML, see, e.g., [30].) One of the advantages of working in XML is the existence of tools such as parsers, visualization tools, development environments, etc.

The instance language OSiL makes up a part of the OSxL project; additional components deal with the solutions returned by the solver, options passed to the solver, etc. (For an overview over the system, see the developers' web site [11].) The original work by Fourer *et al.* [10] covers deterministic linear and nonlinear programs; the current work extends these ideas to the representation of stochastic programs. In addition to the file-based schema, a parallel effort called OSInstance is underway to define in-memory objects to mirror all of the constructs of the OSiL format described below.

The paper is organized as follows: Section 2 sets up the most general stochastic program that the format is prepared to support. This problem is much too general to be of use in this exposition, so a small multistage investment problem is also presented, which will be used as a running example throughout the paper. In Section 3 we quickly review the format for deterministic problems, and in Section 4 we will show how information about the time stages of the problem can be captured.

The next two sections deal with describing the event tree. There are three possibilities. A tree can be given explicitly scenario by scenario, or node by node. The latter allows for the modelling of problems with stochastic dimensions; both options are described in Section 5. It is also possible to define an event tree implicitly, by specifying the distributions of a number of random variables and how these random variables impact on the problem parameters. This option appears in Section 6.

Section 7 shows different ways to express "soft constraints" (that may be violated under extreme circumstances) within the OSiL format. There are two general ways for dealing with soft constraints: adding penalties for constraint violation into the objective, or requiring limits on the risk resulting out of the violation of a constraint. Both are available within OSiL. Section 8 finally draws some conclusions and describes future plans for the system.

2 Stochastic programming problems

The most general problem that can be described in the proposed format takes the following mathematical form:

$$\begin{aligned}
\operatorname*{opt}_{x_0} \quad & \Big[f_0(x_0) + \mathrm{E}_{\xi_1} \Big\{ \operatorname*{opt}_{x_1} \big[f_1(x_0,\xi_1,x_1) + \mathrm{E}_{\xi_2|\xi_1} \{ \operatorname*{opt}_{x_2} f_2(x_0,x_1,\xi_1,\xi_2,x_2) + \dots \\
& \qquad + \mathrm{E}_{\xi_T|\xi_1\dots\xi_{T-1}} [\operatorname*{opt}_{x_T} f_T(x_0,\dots,x_{T-1},\xi_1,\dots,\xi_T,x_T)] \} \big] \Big\} \Big] \\
\text{s.t.} \quad & G_0(x_0) && \sim 0 \\
& \int H_1(x_0,\xi_1) P(d\xi_1) && \sim 0 \\
& G_1(x_0,\xi_1,x_1) && \sim 0 \text{ a.s.} \\
& \int H_2(x_0,x_1,\xi_1,\xi_2) P(d(\xi_1,\xi_2)) && \sim 0 \text{ a.s.} \\
& \quad \vdots \\
& \int H_T(x_0,\dots,x_{T-1},\xi_1,\dots,\xi_T) P(d(\xi_1,\dots,\xi_T)) && \sim 0 \text{ a.s.} \\
& G_T(x_0,\dots,x_{T-1},\xi_1,\dots,\xi_T,x_T) && \sim 0 \text{ a.s.} \\
& x_0 \in X_0, x_1 \in X_1(x_0,\xi_1), \dots, x_T \in X_T(x_0,\dots,x_{T-1},\xi_1,\dots,\xi_T),
\end{aligned} \tag{1}$$

where $X_t(.)$ is a multidimensional box $[l_t(.), u_t(.)]$ of (possibly random) dimension $D_t(.)$. Some components of x_t may further be restricted to integer values only.

The random variables ξ_t and the induced decisions x_t (for $t > 0$) must be adapted to a filtration $\{\emptyset, \Omega\} = \mathcal{F}_0 \subseteq \mathcal{F}_1 \subseteq \ldots \subseteq \mathcal{F}_T \subseteq \mathcal{F}$ of some probability space $(\Omega, \mathcal{F}, P)$, and constraints involving induced decisions x_1, x_2, etc. need hold only almost surely (a.s.), that is, they may be violated on a set of probability measure 0. The dependence of x_t and ξ_{t+1} on the history $\eta^t := (\xi_1, \dots, \xi_t)$ of the data process has been suppressed in order not to overload an already cumbersome notation. For further information on stochastic programs, see, for instance, the texts by Birge and Louveaux [2] and Prékopa [29].

In this formulation the dimension of x_t is allowed to depend on the history η^t, but it is assumed to be bounded. That is, there exists some integer $N > 0$ such that the dimension $D_t(.) \le N$ for every possible realization of the data process $(\xi_1, \dots, \xi_T)$.

The integral constraints $\int H_t(x_0, \dots, x_{t-1}, \eta^t) P(d(\eta^t)) \sim 0$ represent so-called chance constraints [4], integrated chance constraints [19, 32] or higher moment constraints (such as the second-order dominance constraints described in [7]). The symbol $\sim$ is used to allow a mixture of equality and inequality constraints.

This general form of the problem is perhaps not very useful, but it admits a number of special cases that are encountered frequently. We will mention two of them next.

A. If there are no integral constraints and all the distributions are finite, the problem is known as a (nonlinear) *recourse problem.* In this case all the expectations in the objective can be restated as finite sums, and it is possible to reformulate (1) as a large-scale deterministic problem, the so-called *deterministic equivalent.* The linear version of the recourse problem has seen extensive treatment in the literature (see, e.g., [2, 13]).

B. The problem

$$\begin{array}{lll} \underset{x_0}{\text{opt}} & f_0(x_0) & \\ \text{s.t.} & \quad G_0(x_0) & \sim 0 \\ & \displaystyle\int H_1(x_0,\xi_1)P(d\xi_1) & \sim 0 \\ & l_0 \le x_0 \le u_0, x_0 \in R^{n_0} \times Z^{n'_0} & \end{array} \tag{2}$$

is a nonlinear chance-constraint problem (see, e.g., Charnes and Cooper [4]).

We next give a special example of a linear recourse problem taken from the book by Birge and Louveaux [2]. This problem will be used to illustrate most of the stochastic constructs in the OSiL schema. The problem concerns a decision maker who, having available a fixed sum of money B, must choose among several investment opportunities in an attempt to increase his wealth to a target R by the end of the planning horizon. The performance of each investment is random, but the distribution is assumed to be available, based on historic performance. The final wealth is to be maximized, and a penalty is charged whenever the final wealth falls short of the target. At prespecified times the decision maker is allowed to observe the performance of the portfolio and to re-balance it by buying and selling investment instruments. (There are no transaction costs.) Mathematically this problem can be written as follows.

$$\begin{array}{lll} \max & \displaystyle\sum_{s=0}^{S} p_s(w_s - \beta y_s) & \\ \text{s.t.} & \displaystyle\sum_{i=1}^{I} x_{0i} & = B \\ & \displaystyle\sum_{i=1}^{I} \alpha_{1is} x_{0i} - \sum_{i=1}^{I} x_{1is} & = 0, s \in S_1 \\ & \quad \displaystyle\sum_{i=1}^{I} \alpha_{t,i,s} x_{t-1,i,a(s)} - \sum_{i=1}^{I} x_{tis} & = 0, s \in S_t, t = 2,\ldots,T-1 \\ & \qquad \displaystyle\sum_{i=1}^{I} \alpha_{T,i,s} x_{T-1,i,a(s)} + y_s - w_s & = R, s \in S_T \\ & x_{0i}, x_{tis} \ge 0, i = 1,\ldots,I, t = 2,\ldots,T-1, s \in S_{t-1},\ w_s, y_s \ge 0, s \in S_T & \end{array} \tag{3}$$

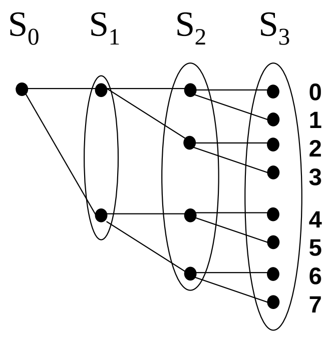

Fig. 2: Event tree for four-stage investment problem.

where x_{tis} is the amount of money invested in investment i during period t under scenario s, B is the original budget, R is the target wealth at the horizon, α_{tis} represents the random return of investment i in stage t under scenario s, β is the penalty (per dollar) of falling short of the target, p_s is the probability of scenario s, y_s and w_s are the amount below and above the target, respectively, under scenario s, and S_t is the set of scenarios that are active in period t, with $S_T = \{0, 1, 2, \ldots, S\}$. When there are multiple scenarios that share the same event history (such as scenarios 0 and 1 in S_2 in figure 2), we represent the corresponding *scenario bundle* [31] by a single scenario (see below). The notation $a(s)$ is used to designate the ancestor scenario of s in the preceding period (see figure 2).

In the book by Birge and Louveaux, $I = 2$, $T = 3$, $B = 55$, $R = 80$, $\beta = 4$, and the returns α_{tis} can take two possible values in each period, depending on whether the economy is good or bad. In a good economy, the returns are $\{1.25, 1.14\}$ for the two investments, and in a bad economy the returns are $\{1.06, 1.12\}$. This leads to a problem with eight equi-probable scenarios ($p_s = 0.125$ for all s) and an event tree that has the form given in figure 2. Horizontal line segments in this figure represent a good economy, while downward sloping line segments represent a bad economy.

A quick word about event trees will round out this section. The central concept of stochastic programming is that of *nonanticipativity*, which says that decisions made today are not allowed to depend on specific realizations of random variables tomorrow; instead all decisions must be based only on information available at the time the decision is made. One convenient way to depict the information structure for finite discrete distributions is through an event tree. Paths through the event tree from the root node in stage 0 to one of the leaves are called *scenarios* and represent the possible evolutions of the data process.

Where two paths are indistinguishable because they share a common history, the corresponding decisions must also coincide. This means, for instance, that in the event tree of figure 2 there are two possible sets of decisions in stage 1, depending on whether the stock market went up or down. Each event in stage 1 can lead to four different branches in the future, but in stage 1 they are indistinguishable from each other. Nonanticipativity can be enforced by "bundling" the four scenarios together and allowing only one set of decision variables — and data — that are shared by the four scenarios. For more on scenario bundles, see [31].

Similarly there are four sets of decisions (i.e., four scenario bundles) in stage 2 and eight possible ways to calculate gains and losses at the end of the planning horizon.

All the scenarios share problem dimensions, sparsity patterns and certain deterministic parameters, such as the coefficient -1 on the variables x_{tis} in equation (3). This information needs to be given only once, and is contained in the "core" information, as explained in more detail in the next section.

In the case of continuous distributions similar considerations hold, although the trees exhibit infinite branching and might more appropriately be called "fans". Nonetheless, we will use the term "event tree" for these objects also.

3 The OSiL format for deterministic problems

In this section we give a quick summary of the deterministic portion of the OSiL format. This material is condensed from [10]. We will illustrate with (3), using one specific scenario. This scenario problem plays the role of what is called the *core problem* in the SMPS format. The core problem may represent any one of the scenarios, or even a related problem, such as the mean value problem in which all random variables are replaced by their mean. The purpose of the core problem is to fix the problem dimensions, to supply the deterministic values that will be shared by all scenarios, and to provide placeholders for random variables to be defined later.

The root node of the OSiL schema is the `osil` node. (We use the terms "tree", "node" and "root") in two different ways in this paper: for the event tree of the stochastic program and for the expression tree of the corresponding XML file. This is unfortunate, but it is unavoidable; we hope that the context makes it clear in each case which tree is being referred to.)

As shown in figure 3, two XML elements combine to form the information in the `osil` node. The `instanceHeader` gives optional information about the problem, such as name, source, plus a description that can be as detailed as the modeller wishes, or can be omitted altogether. Solid boxes in the diagram represent required elements; the small dashed boxes indicate elements that are optional. The three dots following the `osil` and `instanceHeader` nodes indicate that the dependent elements on the right — if present — must appear in the order set out in the diagram.

Figure 4 shows the possible content of the `instanceHeader` for the specific instance of (3) described in the previous section.

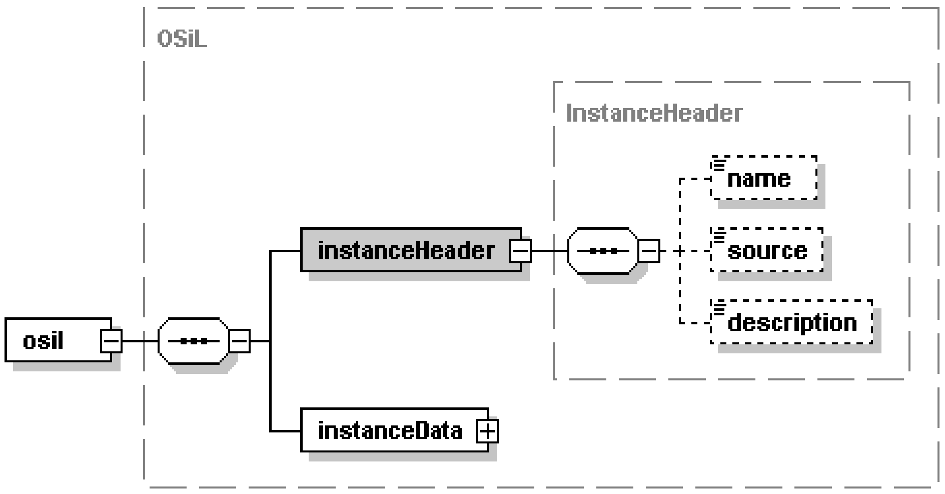

Fig. 3: Instance header information — XML schema.

```
<?xml version="1.0" encoding="UTF-8"?>
<osil xmlns="os.optimizationservices.org"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="os.optimizationservices.org
             http://www.optimizationservices.org/schemas/OSiL.xsd">
    <instanceHeader>
        <name>FinPlan</name>
        <source>Birge and Louveaux, Stochastic Programming</source>
        <description>
            This is the stochastic financial planning problem,
            as given in the book by Birge and Louveaux.
            It has four stages and eight scenarios.
            ...
        </description>
    </instanceHeader>
```

Fig. 4: Instance header for investment problem (2).

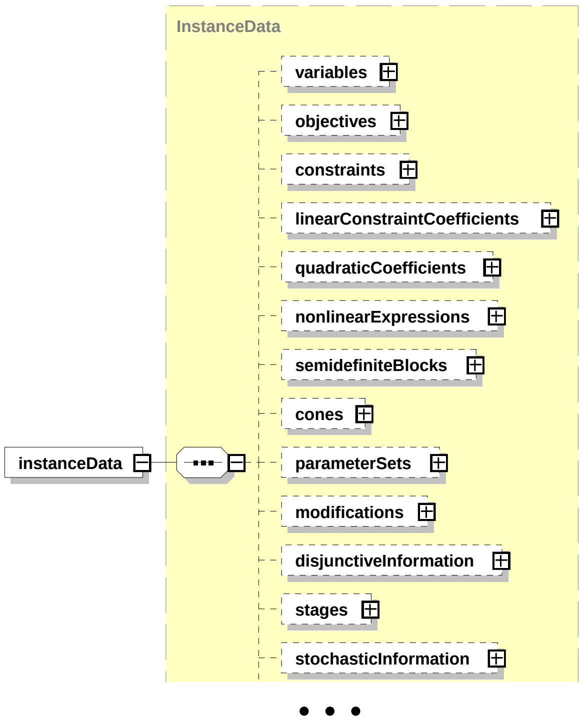

Fig. 5: Instance data — XML schema.

The problem data are described in the `instanceData` element. We look again at the schema first (figure 5). Separate elements describe the variables, constraints, objectives (possibly more than one), nonzero elements of the linear constraint matrix, coefficients of quadratic functions in objectives and constraints (if any), and so on. Figure 6 shows examples of these data items for the Birge and Louveaux problem. It represents one path through the event tree of figure 2, called the "root scenario" (labeled '0' in figure 2). In this scenario the economy is good in every stage. That is, figure 6 represents the deterministic linear program

$$\begin{array}{lllllllllll}
\max & & & & & & & & w - 4y & & \\
\text{s.t.} & x_{01} & + x_{02} & & & & & & & = 55 & \\
 & 1.25\,x_{01} & + 1.14\,x_{02} & - x_{11} & - x_{12} & & & & & = 0 & \quad (4) \\
 & & & 1.25\,x_{11} & + 1.14\,x_{12} & - x_{21} & - x_{22} & & & = 0 & \\
 & & & & & 1.25\,x_{21} & + 1.14\,x_{22} & - w & + y & = 80. &
\end{array}$$

```
<instanceData>
    <variables numberOfVariables="8">
        <var name="Invest0Stocks" type="C" lb="0.0"/>
        <var name="Invest0Bonds"/>
        <var name="Invest1Stocks"/>
        <var name="Invest1Bonds"/>
        <var name="Invest2Stocks"/>
        <var name="Invest2Bonds"/>
        <var name="wealth"/>
        <var name="short"/>
    </variables>
    <objectives numberOfObjectives="1">
        <obj name="expectedWealth" maxOrMin="max"
                                   numberOfObjCoef="2">
            <coef idx="6">1.0</coef>
            <coef idx="7">-4.0</coef>
        </obj>
    </objectives>
    <constraints numberOfConstraints="4">
        <con name="Budget0" lb="55" ub="55"/>
        <con name="Budget1" lb="0"  ub="0"/>
        <con name="Budget2" lb="0"  ub="0"/>
        <con name="Budget3" lb="80" ub="80"/>
    </constraints>
    <linearConstraintCoefficients numberOfValues="14">
        <start>
            <el>0</el>    <el>2</el>    <el>4</el>
            <el>6</el>    <el>8</el>    <el>10</el>
            <el>12</el>   <el>13</el>   <el>14</el>
        </start>
        <rowIdx>
            <el>0</el>    <el>1</el>    <el>0</el>
            <el>1</el>    <el>1</el>    <el>2</el>
            <el>1</el>    <el>2</el>    <el>2</el>
            <el>3</el>    <el>2</el>    <el>3</el>
            <el>3</el>    <el>3</el>
        </rowIdx>
        <value>
            <el> 1</el>    <el>1.25</el>
            <el> 1</el>    <el>1.14</el>
            <el>-1</el>    <el>1.25</el>
            <el>-1</el>    <el>1.14</el>
            <el>-1</el>    <el>1.25</el>
            <el>-1</el>    <el>1.14</el>
            <el>-1</el>    <el>1</el>
        </value>
    </linearConstraintCoefficients>
    ...
</instanceData>
```

Fig. 6: Core scenario data.

4 Stage structure

As stated in Section 2, nonanticipativity in stochastic programs requires careful consideration of the times when the decisions are made and when new information becomes available. This section will explain the issues and how to convey the information within the OSiL format.

When decisions are made over time — even in a deterministic problem — it is important to specify the planning period, which in the present situation is given by the finite discrete set $\mathcal{T} := \{0, 1, \ldots, T\}$ (see equation (1)). The stages of the problem form a subset of the time structure. Each stage in a stochastic program contains decision points and event points. Decision points that make use of the same information (i.e., without any further realizations of random events being obtained in the interim) can be aggregated into the same stage; similarly, multiple realizations of random variables that are observed without any decisions being taken in the meantime, can be collected into just a single event point. Each stage, therefore, contains one set of decisions, x_t, and one set of random variables, ξ_t. The user must decide whether the random variables precede the decisions or follow them. There are advantages and disadvantages to either setup (see [14]), and OSiL supports both.

But time structure is also useful in deterministic problems, for instance in dynamic programming. For this reason OSiL does not distinguish between periods and stages. In particular, no check is performed to ensure that there is stochastic information present for every stage defined in the `stages` element. The user is also free to specify whether decisions precede events within a period or follow them (see Section 5).

The function of the `stages` element is to specify the number of stages (or periods) in the problem and to define for each variable and constraint previously introduced (see Section 3) to which stage it belongs. There are two mechanisms, depending on the appearance of the `variables` and `constraints` elements. If variables and constraints were given in temporal order, then all that is needed is to record for each stage the number of variables it contains and the index of the first and last variable. (In fact, the variable indices could be inferred from the other information.) On the other hand, if the variables were originally given in arbitrary order, then an explicit list is needed, giving for each variable and constraint explicit information on the stage to which it belongs.

Figure 7 shows the OSiL schema element `stages` that caters to both possibilities. The corresponding segment of the Birge and Louveaux problem is given in figure 8. Note that, although both variable and constraint sections were originally set up in temporal order (see figure 6), we have chosen to illustrate the unordered option for stages 2 and 3.

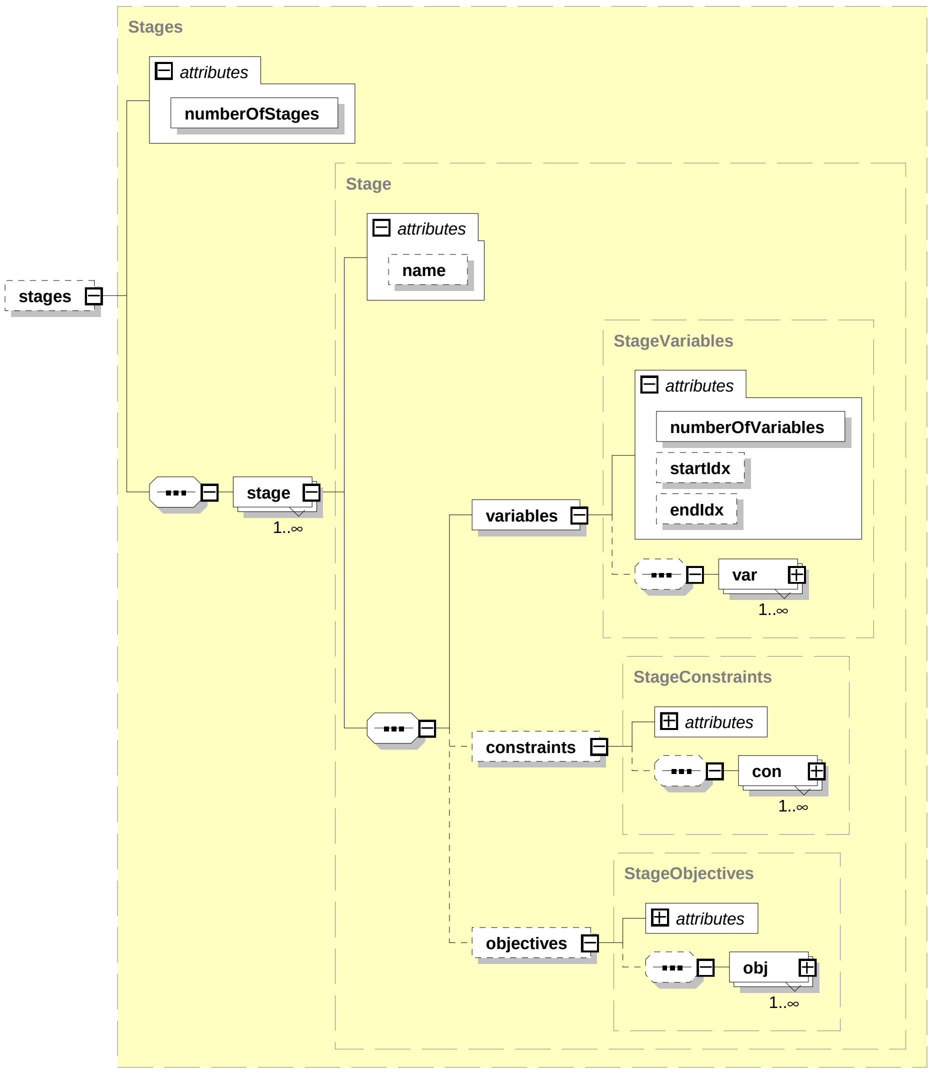

Fig. 7: Time stages — XML schema.

```
<stages numberOfStages="4">
    <stage name="stage_0">
        <variables numberOfVariables="2" startIdx="0" endIdx="1"/>
        <constraints numberOfConstraints="1" startIdx="0"
endIdx="0"/>
    </stage>
    <stage name="stage_1">
        <variables numberOfVariables="2" startIdx="2" endIdx="3"/>
        <constraints numberOfConstraints="1" startIdx="1"
endIdx="1"/>
    </stage>
    <stage name="stage_2">
        <variables numberOfVariables="2">
            <var idx="4"/>
            <var idx="5"/>
        </variables>
        <constraints numberOfConstraints="1">
            <con idx="2"/>
        </constraints>
    </stage>
    <stage name="stage_3">
        <variables numberOfVariables="2">
            <var idx="6"/>
            <var idx="7"/>
        </variables>
        <constraints numberOfConstraints="1">
            <con idx="3"/>
        </constraints>
    </stage>
</stages>
```

Fig. 8: Stage information for investment problem.

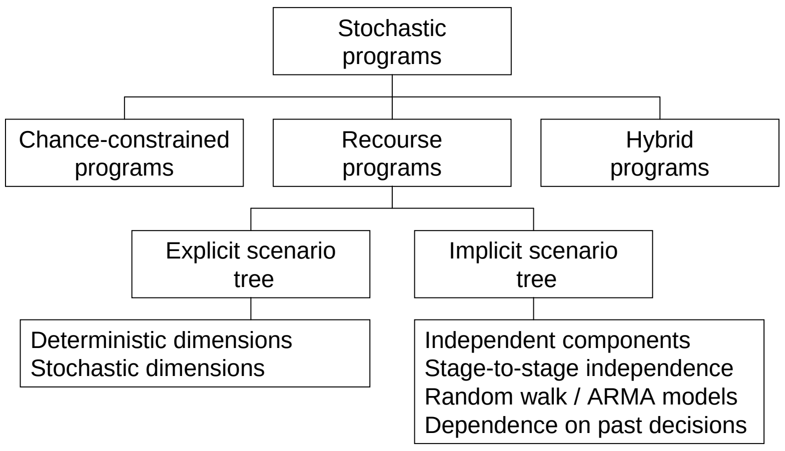

Fig. 9: A taxonomy of stochastic programming models.

5 Explicit event trees

The major effort in a stochastic program is (naturally) the representation and generation of the stochastic data. Somehow the event tree must be generated, and with it the data contained in it. In particular, the stochastic representation does not give the full deterministic equivalent (3) explicitly. Rather, it contains all the information necessary so that another layer of software can create the deterministic equivalent or any portion of it (node, path, subtree, etc.) whenever required by an algorithm, for instance the nested Benders decomposition of [13].

Since frequently only a small portion of the data is stochastic, there is considerable redundancy, which should be exploited. However, the optimal problem representation will depend on the nature of the uncertainty, and we will first give a schematic overview of the different problem types (see figure 9).

Different types of models permit different ways of constructing the event tree in OSiL. Figure 10 gives the high-level overview of the available options. The attribute `decisionEventSequence` allows the user to specify whether decisions follow events in a stage (as in figure 12) or precede them.

If the distributions of all random parameters are finite, and particularly if they exhibit dependence from one stage to the next, an explicit approach works best in practice. A scenario-based format is used when the problem dimensions are deterministic. In this case, the deterministic problem set up in earlier data elements (`variables`, `constraints`, `objectives`, `linearConstraintCoefficients`, etc.) can be used as a blueprint for each of the scenario problems and recording for each scenario the period when it becomes active along with the parent scenario from which it branches.

For the investment problem of Section 2, one possible scheme is the following:

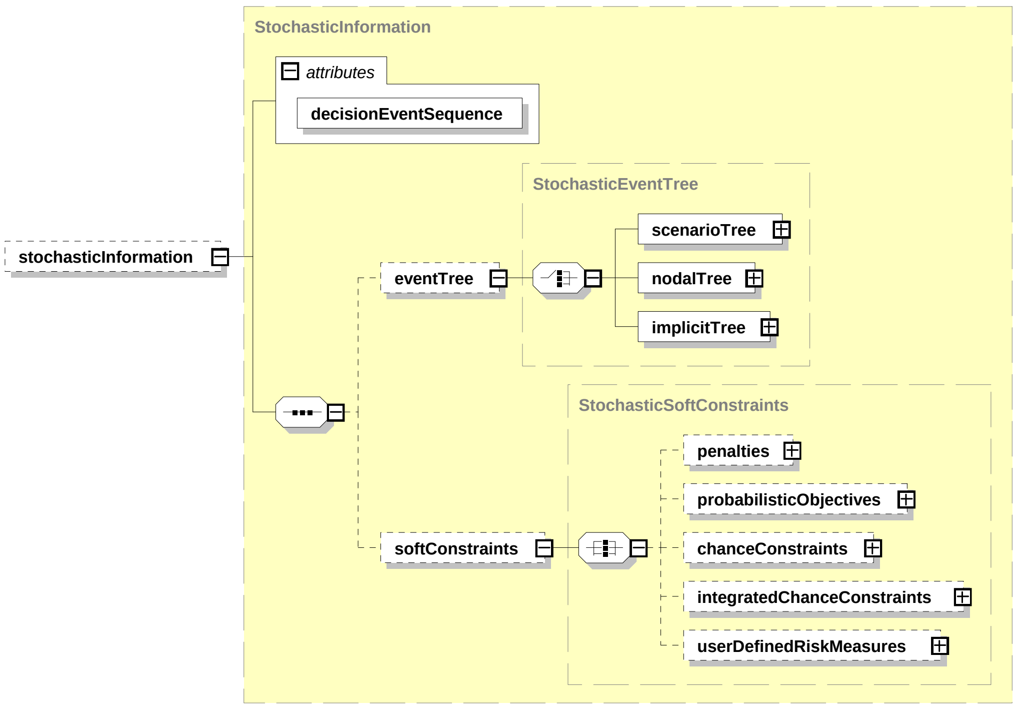

Fig. 10: Stochastic models — XML schema.

Scenario	First stage	Parent
0	stage_0	—
1	stage_3	0
2	stage_2	0
3	stage_3	2
4	stage_1	0
5	stage_3	4
6	stage_2	4
7	stage_3	6

Table 1. Parent scenarios for the investment problem

Scenario 0 is the root scenario, and in each case the base or parent scenario is taken as the member of the bundle that has the lowest number. The data process is characterized by multiple inheritance: Any data item not specifically mentioned is assumed to be the same as the corresponding data item in the parent scenario, or in the parent's parent, etc. The root scenario inherits items not specifically mentioned from the data in the core file. Only those data need to be recorded that differ from the parent scenario (or from any other convenient reference scenario). The OSiL schema element that accomplishes this is called the `scenarioTree`, with the definition shown in Figure 11.

The data tend to be repetitive, so the example in figure 12 shows only the first three scenarios of the investment problem. The data for the root scenario (scenario 0 in figure 2) have already been given in the deterministic data section, hence there is no need for repetition.

The next scenario (scenario 1) branches from it in stage 3. It represents a bad economy in stage 3, hence the coefficients 1.06 and 1.12 replace, respectively, the values 1.25 and 1.14 in the last constraint. Taken as a deterministic LP, this scenario therefore is represented by

$$\begin{array}{lllllllll}
\max & & & & & & w - 4y & & \\
\text{s.t.} & x_{01} + x_{02} & & & & & & = 55 & \\
& 1.25\,x_{01} + 1.14\,x_{02} & - x_{11} - x_{12} & & & & & = 0 & (5)\\
& & 1.25\,x_{11} + 1.14\,x_{12} & - x_{21} - x_{22} & & & & = 0 & \\
& & & 1.06\,x_{21} + 1.12\,x_{22} & - w & + y & & = 80. &
\end{array}$$

Scenario 2 branches from scenario 0 in stage 2 and represents a bad economy in stage 2 and a good economy in stage 3. (Since stage 3 information is shared with scenario 0, there is no need to repeat it here.) The full linear program corresponding

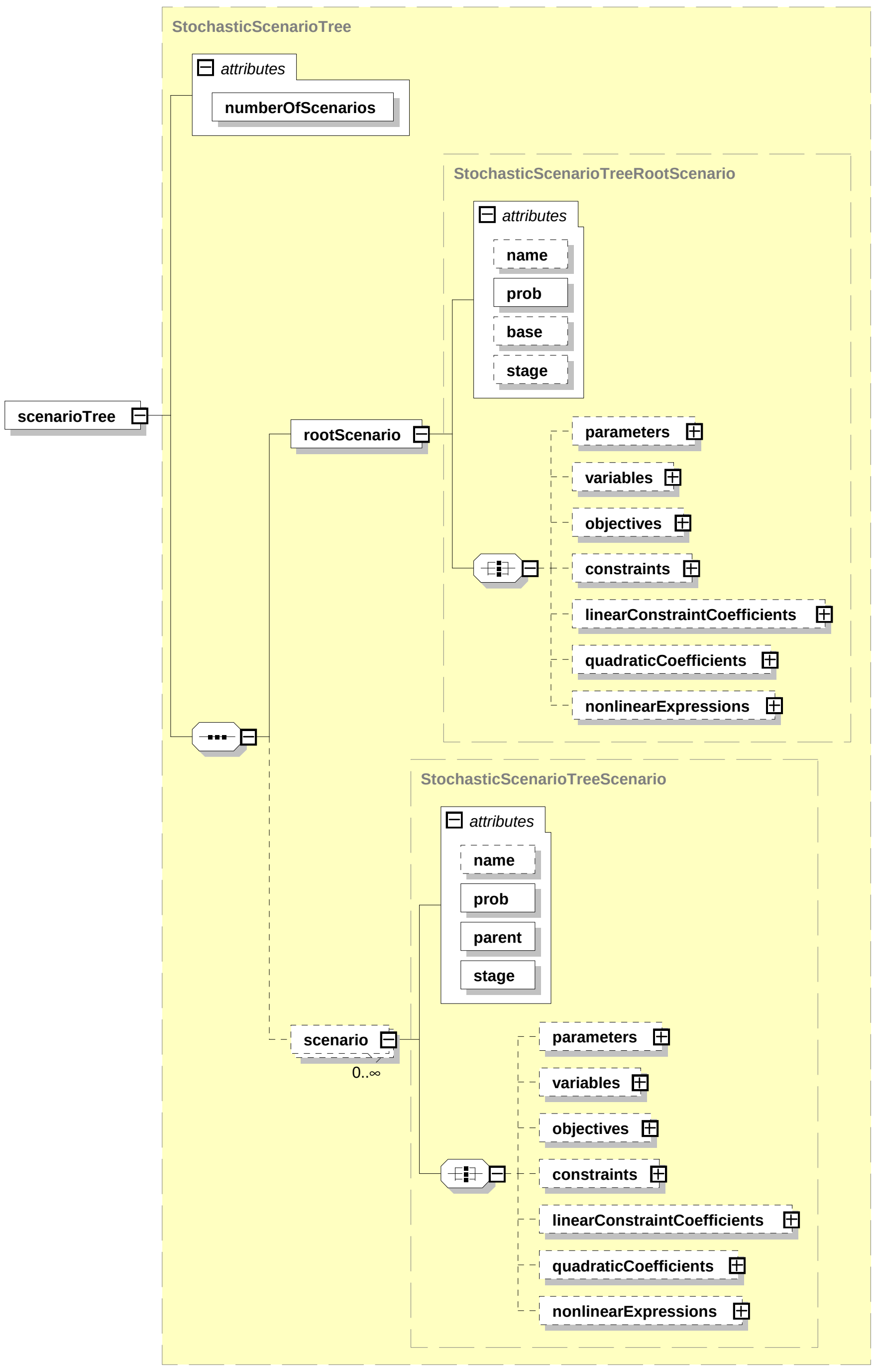

Fig. 11: Scenario tree — XML schema.

```
<stochasticInformation decisionEventSequence="decisionAfterEvent">
  <eventTree>
    <scenarioTree numberOfScenarios="8">
        <rootScenario prob="0.125"/>
        <scenario stage="3" prob="0.125" parent="0">
            <linearConstraintCoefficients numberOfValues="2">
                <el rowIdx="3" colIdx="4">1.06</el>
                <el rowIdx="3" colIdx="5">1.12</el>
            </linearConstraintCoefficients>
        </scenario>
        <scenario stage="2" prob="0.125" parent="0">
            <linearConstraintCoefficients numberOfValues="2">
                <el rowIdx="2" colIdx="2">1.06</el>
                <el rowIdx="2" colIdx="3">1.12</el>
            </linearConstraintCoefficients>
        </scenario>
        ...
    </scenarioTree>
  </eventTree>
</stochasticInformation>
```

Fig. 12: Sample scenario tree.

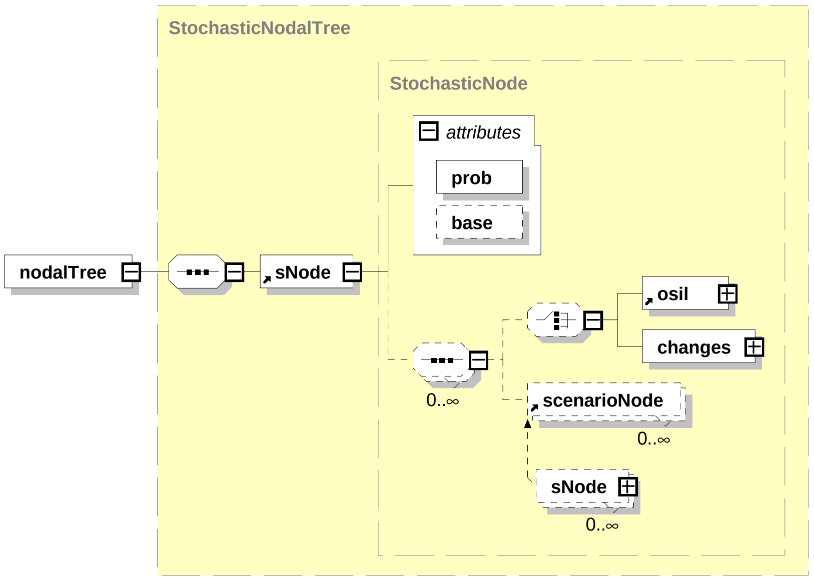

Fig. 13: Node-by-node tree — XML schema.

to this scenario is

$$
\begin{array}{llllllll}
\max & & & & & w - 4y & & \\
\text{s.t.} & x_{01} + x_{02} & & & & & = 55 & \\
& 1.25\,x_{01} + 1.14\,x_{02} & - x_{11} - x_{12} & & & & = 0 & (6)\\
& & 1.06\,x_{11} + 1.12\,x_{12} & - x_{21} - x_{22} & & & = 0 & \\
& & & 1.25\,x_{21} + 1.14\,x_{22} & - w & + y & = 80. &
\end{array}
$$

If the problem size is stochastic, then the scenario-wise description is not available. Instead the event tree and with it the deterministic equivalent problem can be built node for node in the `nodalTree` element, as shown in figure 13. For each node in the event tree it is possible to define the data explicitly, using an `osil` node in recursive fashion. This allows complete freedom in the specification of the tree, including problem dimension, linear and nonlinear functions, etc.

It is also possible to reference data from a reference node and to record changes, similarly to the `scenarioTree` element. This assumes that the current node and the reference node have the same dimensions.

We omit an explicit example of the `nodalTree` element. The `changes` option is very similar to figure 12 and provides no new insights, while the explicit construction is extremely verbose and not recommended when the problem dimensions are deterministic, as they are for the investment problem.

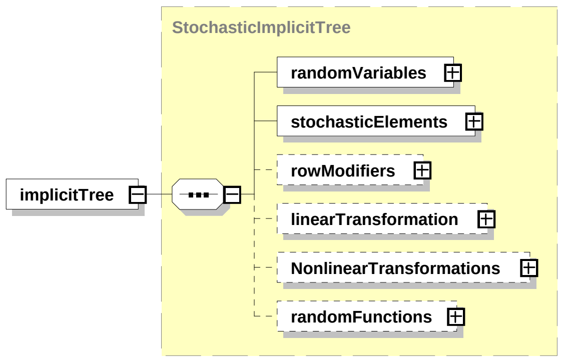

Fig. 14: Implicit event trees— XML schema.

6 Implicit event trees

When the distributions of the random variables are independent of each other, or when they are subject to random processes that are influenced by random variables with period-to-period independence, then the event tree can also be built implicitly. Figure 14 shows the general OSiL schema. What is required in this case is a specification of the random variables and a set of transformations that link these random variables to the coefficients and parameters of the model.

The schema for specifying random variables is depicted in figure 15. There are several univariate and multivariate distributions defined, such as the uniform, normal, lognormal, beta, gamma distributions, etc., and in addition the user can specify any arbitrary distributions via nonlinear functions.

The investment problem in Section 2 has three simple two-stage distributions, and we illustrate two ways to specify them. In figure 16, the distributions of the two returns in stages 1 and 2 are given as explicit two-point discrete random vectors, while the distribution in stage 3 is given as a Bernoulli random variable (or equivalently, as binomial(N, p) with $N = 1$ and $p = 0.5$). The two-point distribution of the random vector $r = (r_1, r_2)$ can then be recovered by writing

$$r = r_0 + \xi_3 * \Delta r \tag{7}$$

where r_0 and Δr are fixed values and ξ_3 represents the Bernoulli random variable. In the investment example, $r_0 = (1.25, 1.14)$ and $\Delta r = (-0.19, -0.02)$. If the Bernoulli random variable ξ_3 takes a value of 0, this yields the realization $(1.25, 1.14)$ for the returns in stage 3, and if the Bernoulli random variable takes value 1, the returns are $(1, 06, 1.12)$. Note that equation (7) decomposes a multi-dimensional random vector into a single one-dimensional factor.

It is important to note that the random variables have not yet been tied to any stochastic problem parameters, which allows for considerable flexibility. The linkage

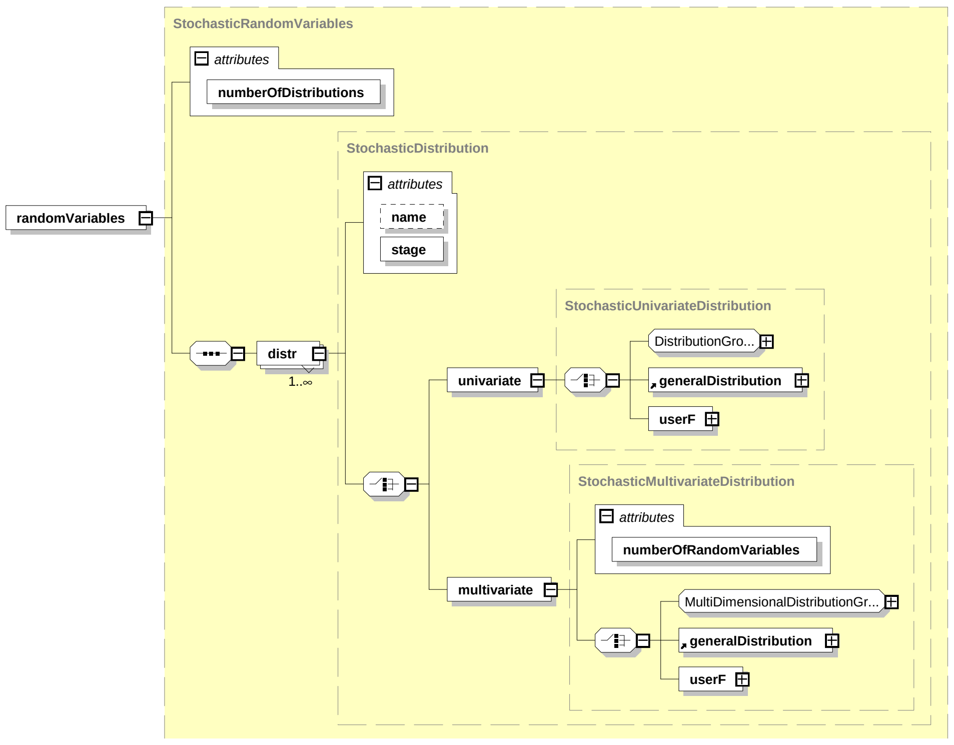

Fig. 15: Distributions of random variables — XML schema.

```
<randomVariables numberOfDistributions="3">
    <distr stage="2">
        <multivariate numberOfRandomVariables="2">
            <multivariateDiscrete>
                <scenario prob="0.5">
                    <el>1.25</el>
                    <el>1.14</el>
                </scenario>
                <scenario prob="0.5">
                    <el>1.06</el>
                    <el>1.12</el>
                </scenario>
            </multivariateDiscrete>
        </multivariate>
    </distr>
    <distr stage="3">
        <multivariate numberOfRandomVariables="2">
            <multivariateDiscrete>
                <scenario prob="0.5">
                    <el>1.25</el>
                    <el>1.14</el>
                </scenario>
                <scenario prob="0.5">
                    <el>1.06</el>
                    <el>1.12</el>
                </scenario>
            </multivariateDiscrete>
        </multivariate>
    </distr>
    <distr stage="4">
        <multivariate numberOfRandomVariables="2">
            <multivariateDiscrete>
                <scenario prob="0.5">
                    <el>1.25</el>
                    <el>1.14</el>
                </scenario>
                <scenario prob="0.5">
                    <el>1.06</el>
                    <el>1.12</el>
                </scenario>
            </multivariateDiscrete>
        </multivariate>
    </distr>
</randomVariables>
```

Fig. 16: Univariate and multivariate distributions.

between the random variables and the stochastic problem parameters is made by specifying explicit transformations. The schema for these transformations is given in figure 17.

This separation of random variables and stochastic parameters allows for compact representation of complicated random processes, for instance ARMA or GARCH processes, with comparable ease. In the current case we have simple linear transformations, which in the case of the random variables in stage 1 and 2 are just identities.

For the investment problem (3) the collection of all transformations can be given as a single affine transformation

$$\begin{pmatrix} r_{11} \\ r_{12} \\ r_{21} \\ r_{22} \\ r_{31} \\ r_{32} \end{pmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & -0.19 \\ 0 & 0 & 0 & 0 & -0.02 \end{bmatrix} \begin{pmatrix} \xi_{11} \\ \xi_{12} \\ \xi_{21} \\ \xi_{22} \\ \xi_{3} \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1.25 \\ 1.14 \end{pmatrix}.$$

There are only five random variables (or components of random vectors), because the last two stochastic elements are generated from a single random variable using equation (7).

Figure 18 shows the content of the element `stochasticTransformations`.

7 Soft constraints

It is not always possible to insist that all constraints be satisfied under all possible realizations of the random variables; occasionally there may be a need to relax a constraint a little. This is commonly termed a "soft constraint", and there are two general ways of dealing with such a constraint: One could impose a penalty term (that is added to the objective) and "tax" the violation of a constraint. Since violating this constraint is expensive for the decision maker, there is an incentive to satisfy the constraint "most of the time". The other option is to limit the exposure to risk. For instance, one could require that the constraint be satisfied in at least 95% of all possible situations, or that the expected constraint violation be small, or that some other moment constraint be satisfied.

OSiL contains provisions for both approaches, as illustrated in figure 19.

The objective function of the investment problem is essentially a penalty term: falling short of the target is penalized, while exceeding the target earns a small reward. This kind of objective could be expressed in OSiL as shown in figure 20. Since the penalty function is linear for each type of constraint violation, the problem is technically called a "simple recourse problem". If this formulation of the problem

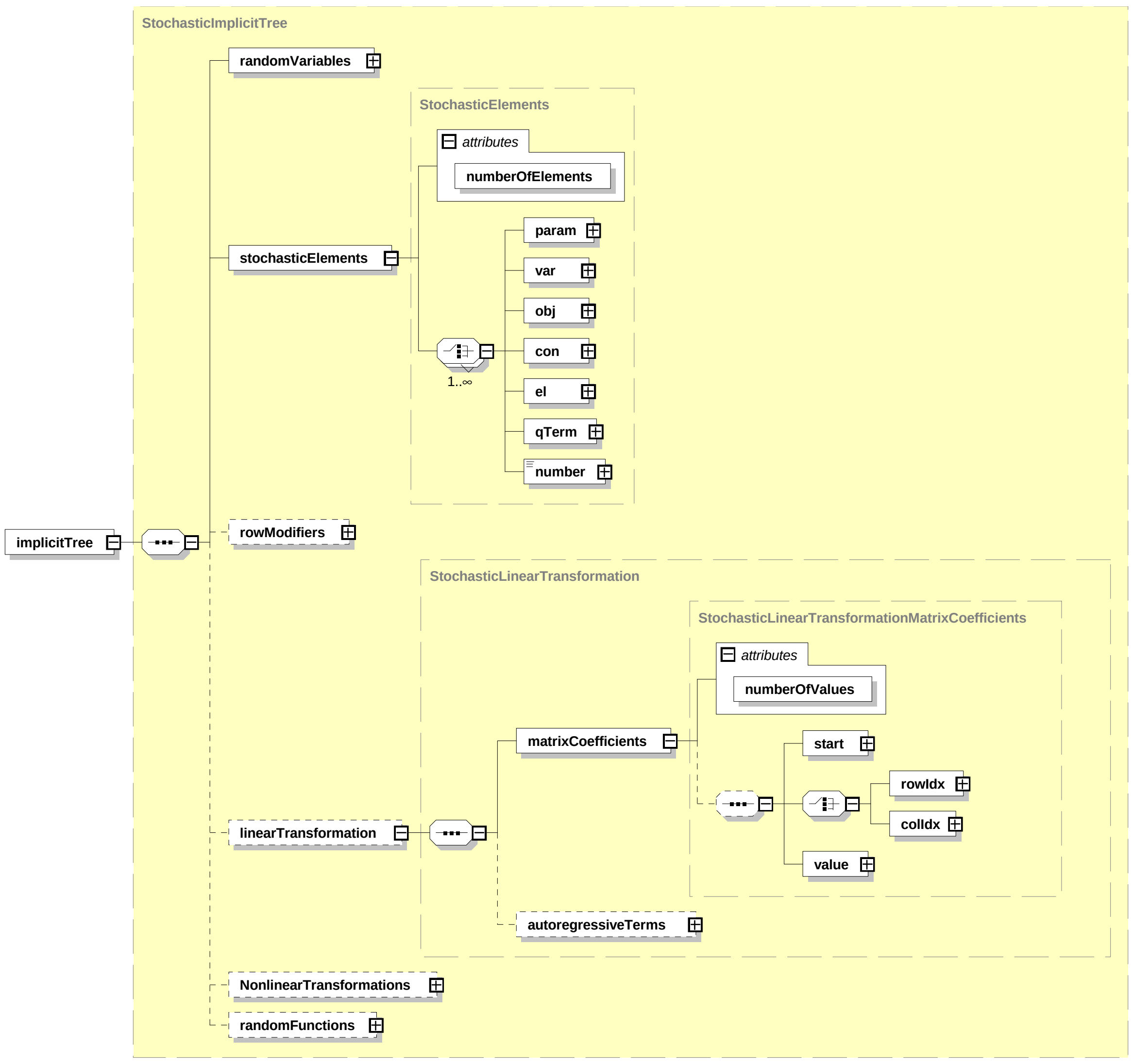

Fig. 17: Transformations of random variables — XML schema.

```
<stochasticElements numberOfElements="6">
    <el rowIdx="1" colIdx="0"/>
    <el rowIdx="1" colIdx="1"/>
    <el rowIdx="2" colIdx="2"/>
    <el rowIdx="2" colIdx="3"/>
    <el rowIdx="3" colIdx="4" baseValue="1.25"/>
    <el rowIdx="3" colIdx="5" baseValue="1.14"/>
</stochasticElements>
<linearTransformation>
    <matrixCoefficients numberOfValues="6">
        <start>
            <el>0</el>
            <el>1</el>
            <el>2</el>
            <el>3</el>
            <el>4</el>
            <el>5</el>
            <el>6</el>
        </start>
        <rowIdx>
            <el>0</el>
            <el>1</el>
            <el>2</el>
            <el>3</el>
            <el>4</el>
            <el>5</el>
        </rowIdx>
        <value>
            <el>1.0</el>
            <el>1.0</el>
            <el>1.0</el>
            <el>1.0</el>
            <el>-0.19</el>
            <el>-0.02</el>
        </value>
    </matrixCoefficients>
</linearTransformation>
```

Fig. 18: Linear transformations for investment problem (3).

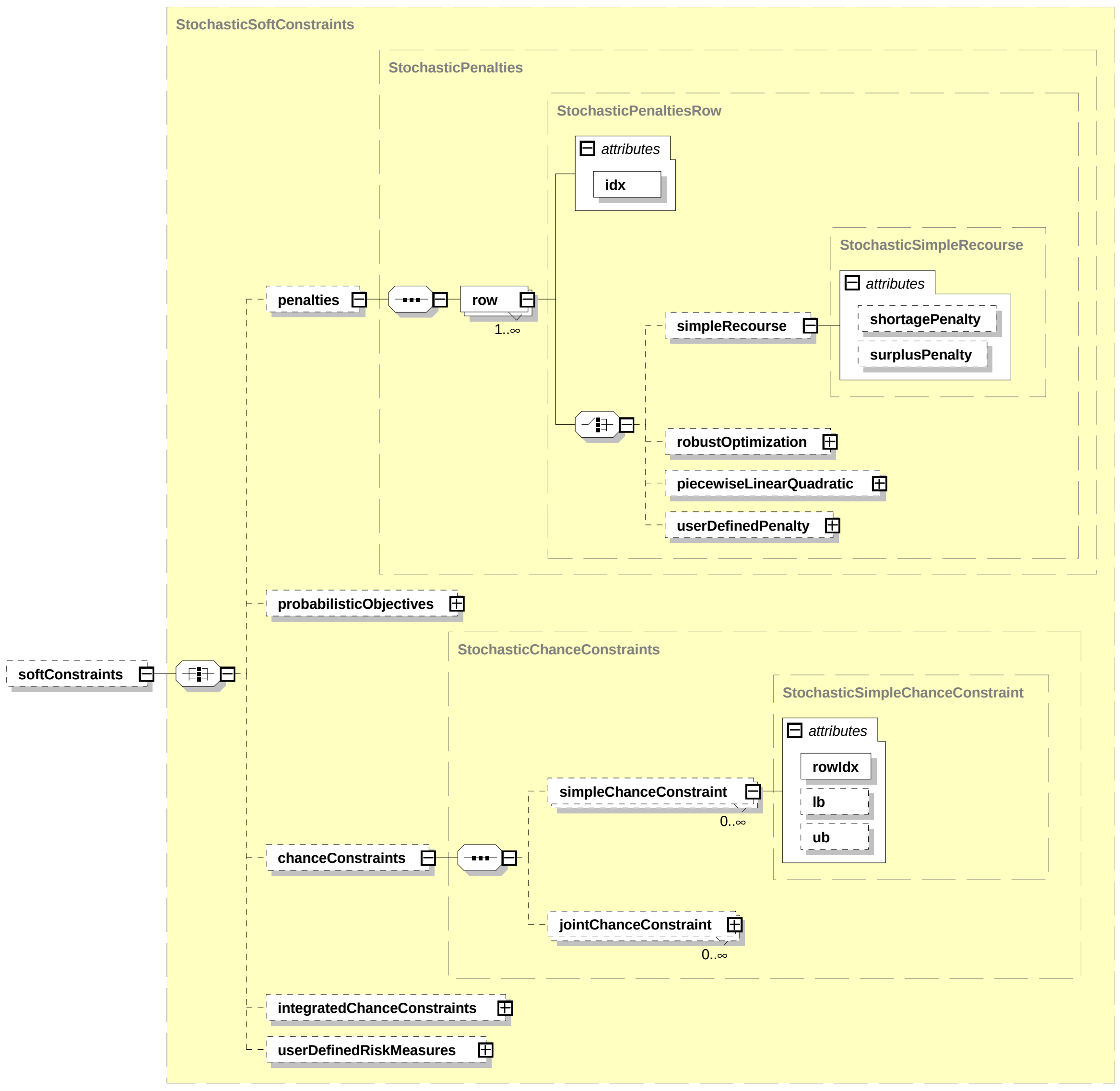

Fig. 19: Soft constraints — XML schema.

```
<softConstraints>
    <penalties>
        <row idx="3">
            <simpleRecourse surplusPenalty="1" shortagePenalty="-4"/>
        </row>
    </penalties>
</softConstraints>
```

Fig. 20: Soft constraints — simple recourse.

is chosen, there will be only six decision variables instead of eight, and the two objective coefficients in figure 6 will have to be removed.

8 Conclusions and future plans

This paper has shown the design of a very general instance description for multistage stochastic nonlinear programs. The stochastic format complements and extends earlier endeavours towards a unified format for general deterministic optimization problems and is part of an on-going development of XML schemas for virtually every aspect of the model development life cycle.

The format can handle continuous as well as discrete distributions, stochastic problem dimensions, ARMA and GARCH processes, arbitrary penalties for soft constraints, linear and nonlinear objectives and constraints, probabilistic constraints and other risk measures.

Only a small portion of the format's features has been illustrated in this paper, mostly due to length restrictions. Additional papers as well as a web site are planned in order to explain other aspects such as stochastic problem dimensions, general user-defined probability distributions, soft constraints, nonlinear problems, etc.

At present only finite discrete time periods are supported, but we have plans to extend the schema to include infinite horizons as well as continuous time decision problems.

Future development will also include schemas for capturing solver output and solver communication, readers and writers, internal data objects for storing the problem instances, and a library of test problems illustrating the many features of the format.

Acknowledgements

The first and third authors were supported by grant DMI-0322580 from the U.S. National Science Foundation, and by a gift from the IBM Faculty Awards program. The second author was supported in part by a grant from the Natural Sciences and Engineering Research Council of Canada (NSERC).

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 multiperiod stochastic linear programs", *COAL Newsletter #17*, pp. 1–19, 1987.

[2] J.R. Birge and F. Louveaux, *Introduction to Stochastic Programming*, Springer Series in Operations Research, Springer Verlag, New York, 1997.

[3] A. Brooke, D. Kendrick and A. Meeraus, *GAMS — A User's Guide*, The Scientific Press, Redwood City, CA, 1988.

[4] A. Charnes and W.W. Cooper, "Chance-constrained programming", *Management Science* **5** (1959) 73–79.

[5] A.R. Conn, N.I.M. Gould and P.L. Toint, "The SIF Reference Document", world-wide web document http://www.numerical.rl.ac.uk/lancelot/sif/sif.html, accessed 13 July 2006.

[6] W. J. Cook, "MPS Format ",world-wide web document http://www2.isye.gatech.edu/ wcook/qsopt/hlp/ff_mps_format.htm, accessed 6 September 2006.

[7] D. Dentcheva and A. Ruszczyński, "Portfolio optimization with stochastic dominance constraints", *Journal of Banking and Finance* **30**, No. 2 (2006) 433–451.

[8] J. Edwards, "A proposed standard input format for computer codes which solve stochastic programs with recourse", in: Yu. Ermoliev and R.J-B Wets (eds.), *Numerical Techniques for Stochastic Optimization*, Springer Series in Computational Mathematics, Vol. 10, Springer Verlag, Berlin, 1988, pp. 215–227.

[9] R. Fourer, D.M. Gay and B.W. Kernighan, *AMPL — A Modeling Language for Mathematical Programming* (2nd ed.), Brooks/Cole—Thomson Learning, Pacific Grove, CA, 2003.

[10] R. Fourer, J. Ma and R.K. Martin, "Optimization Services Instance Language (OSiL), a General-Purpose Instance Representation for Optimization Problems", Working paper, Department of Industrial Engineering and Management Science, Northwestern University, Evanston, Illinois.

[11] R. Fourer, J. Ma and R.K. Martin, "Optimization Services (OS) Overview ", world-wide web document http://gsbkip.chicagogsb.edu/os/os_overview.html, accessed 13 November 2006.

[12] H.I. Gassmann, "The SMPS format for stochastic linear programs", world-wide web document http://myweb.dal.ca/gassmann/smps2.htm, accessed 13 July 2006.

[13] H.I. Gassmann, "MSLiP: An algorithm for the multistage stochastic linear programming problem", *Mathematical Programming* **47** (1990) 407–423.

[14] H.I. Gassmann and A. Prékopa, "On stages and consistency in stochastic programming", *Operations Research Letters* **33** No. 2 (March 2005), 171–175.

[15] H.I. Gassmann and E. Schweitzer, "A comprehensive input format for stochastic linear programs", *Annals of Operations Research* **104** (2001) 89–125.

[16] D.M. Gay, "Electronic mail distribution of linear programming test problems", Numerical Analysis Manuscript 86-0, AT&T Bell Laboratories, Murray Hill, NJ, 1986.

[17] B.V. Halldórsson, E.S. Thorsteinsson and B. Kristjánsson, "A Modeling Interface to Non-Linear Programming Solvers - An instance: xMPS, the extended MPS format", Carnegie Mellon University Mathematical Sciences Department Working Paper, 2001, available as world-wide-web document http://www.mmedia.is/esth/papers/xmps-2000_022000.pdf, accessed 13 November 2006.

[18] W. Hock and K. Schittkowski, *Test Examples for Nonlinear Programming Codes*, Lecture Notes in Economics and Mathematical Systems, Vol. 187, Springer Verlag, 1981.

[19] W.K. Klein Haneveld, *Duality in Stochastic Linear and Dynamic Programming*, Lecture Notes in Economics and Mathematical Systems, Vol. 274, Springer-Verlag, Berlin, 1986.

[20] D. Klingman, A. Napier and J. Stutz, "NETGEN: A program for generating large scale capacitated assignment, transportation, and minimum cost flow network problems", *Management Science* **20** (1974) 814–821.

[21] LPSolve, “MPS file format”, world-wide web document http://lpsolve.sourceforge.net/5.1/mps-format.htm accessed 6 September 2006.

[22] J. Ma, “Optimization Services (OS)”, PhD Thesis, Industrial Engineering and Management Sciences, Northwestern University, 2005. Available as world-wide web document http://gsbkip.chicagogsb.edu/os/publications/Thesis2005.pdf, accessed 6 September 2006.

[23] J. Ma, “Welcome to the Official Optimization Services (OS) Home ”, world-wide web document, http://www.optimizationservices.org/, accessed 23 August 2006.

[24] MOSEK ApS, “The MPS file format”, world-wide web document http://www.mosek.com/products/3/tools/doc/html/toolbox/node10.html, accessed 13 July 2006.

[25] B.A. Murtagh and M.A. Saunders, MINOS 5.1 User’s Guide, Technical Report SOL 83-20R, Systems Optimization Laboratory, Stanford University, 1987 (revised).

[26] J.L. Nazareth, *Computer Solution of Linear Programs*, Oxford University Press, New York, 1987.

[27] Netlib.org, “Netlib linear programming test set”, world-wide web document http://www.netlib.org/lp/index.html, accessed 13 July 2006.

[28] Pintér Consulting Services, “LGO: A Model Development System for Continuous Global Optimization”, User’s Guide, Halifax, Nova Scotia, Canada, 2002.

[29] A. Prékopa, *Stochastic Programming*, Kluwer Academic Publishers, Dordrecht/ Boston/London, 1995.

[30] Refsnes Data, “Introduction to XML”, world-wide web document http://www.w3schools.com/xml/xml_whatis.asp, accessed 13 November 2006.

[31] R.T. Rockafellar and R.J-B Wets, “Scenario and policy aggregation in optimization under uncertainty”, *Mathematics of Operations Research* **16** (1991) 119–147.

[32] R.T. Rockafellar and S. Uryasev, “Optimization of Conditional Value-At-Risk”, *The Journal of Risk* **2** (2000) 21–41.

[33] K. Schittkowski, *More Test Examples for Nonlinear Programming*, Lecture Notes in Economics and Mathematical Systems, Vol. 282, Springer Verlag, 1987.

[34] K. Schittkowski, "Test Problems for Nonlinear Programming — User's Guide", world-wide web document http://www.uni-bayreuth.de/departments/math/~kschittkowski/tpnp.htm, accessed 13 July 2006.

[35] E.S. Thorsteinsson, "xMPS, the Extended MPS Format for Non-Linear Programs", Technical Report 99-224, Carnegie Mellon University Mathematical Sciences Department, December 1999, available as world-wide-web document http://www.mmedia.is/esth/papers/xmpstech-1999_121999.pdf, accessed 13 November 2006.

[36] R.S. Tsay, *Analysis of Financial Time Series*, Wiley Publishers, 2002.

[37] R.J. Vanderbei, "LOQO User's Manual — Version 4.05", Technical report ORFE-99-77, Department of Operations Research and Financial Engineering, Princeton University, Princeton, New Jersey. Also available as world-wide web document http://www.princeton.edu/ rvdb/tex/loqo/loqo405.pdf.

[38] Wolfram Research, Inc., "Multivariate ARMA Models", world-wide web document http://documents.wolfram.com/applications/timeseries/UsersGuidetoTimeSeries/1.2.5.html, accessed 13 July 2006.

[39] Ziena Optimization Inc, "Knitro 5.0 Interfaces", world-wide web document http://www.ziena.com/interfaces.htm, accessed 13 July 2006.