A STANDARD INPUT FORMAT FOR MULTIPERIOD STOCHASTIC LINEAR PROGRAMS

J.R. Birge

Department of Industrial and Operations Engineering
University of Michigan, Ann Arbor, USA

M.A.H. Dempster

Department of Mathematics, Statistics and Computing Sciences
Dalhousie University, Halifax, Canada
and Balliol College, Oxford, England

H.I. Gassmann

School of Business Administration
Dalhousie University, Halifax, Canada

E.A. Gunn

Department of Industrial Engineering
Technical University of Nova Scotia, Halifax, Canada

A.J. King

International Institute for Applied Systems Analysis
Laxenburg, Austria

S.W. Wallace

Chr. Michelsen Institute, Bergen, Norway

ABSTRACT

Data conventions for the automatic input of multiperiod stochastic linear programs are described. The input format is based on the MPSX standard and is designed to promote the efficient conversion of originally deterministic problems by introducing stochastic variants in separate files. A flexible "header" syntax generates a useful variety of stochastic dependencies. An extension using the NETGEN format is proposed for stochastic network programs.

0. Introduction

The desire to solve stochastic optimization problems as more realistic models than deterministic formulations for decision making began in the early days of linear programming, see Dantzig [**3**] for example. As evidenced by recent volumes, Prékopa and Wets [**14**] and Ermoliev and Wets [**5**], there is a wide body of active research into numerical algorithms. However, these efforts have tended to be directed towards specialized types of problems resulting in a potpourri of numerical codes, problem formulations and data sets that are not comparable or even mutually compatible. The development of general purpose codes suitable for industrial application is hampered by the lack of a comprehensive set of test problems in a standard format to compare performance and challenge competitors. In this paper we present our ideas on what we hope will become the nucleus of a standard problem format for multistage stochastic linear programs.

The present contribution is an extension, simplification, and refinement of the ideas in Edwards, et al. [**4**]. The spirit of the two proposals is identical: (a) to use the MPSX convention for file formats; (b) to allow originally deterministic problems to be changed to stochastic versions; (c) to permit a wide variety of stochastic dependencies; and (d) to allow new options to be added. Two key aspects of the original proposal are retained—the use of separate files to specify parameters related to random variables and the use of "header lines" to express and organize the relevant parameters.

After a statement of the problem in the form we shall consider it, we present in Section 2 our proposed input format. In Section 3 we present an extension of the standard format to cover multistage stochastic network problems using the NETGEN format [**10**], and in the final section we set out our recommendations for mixed stochastic network and linear programs.

The collection of test problems is an ongoing enterprise in which readers are encouraged to participate. Some examples, augmenting those already presented in King [**9**] and illustrating the standard problem format, are available in computer accessible form.

1. Problem Statement

A general form of the multiperiod stochastic linear program is:

$$\begin{aligned} \text{minimize} \quad & c_1 x_1 + E_2 Q_2(x_1) \\ \text{subject to} \quad & x_1 \in I\!R^n \\ & \ell_1 \le x_1 \le u_1 \\ & A_1 x_1 = b_1, \end{aligned}$$

where the functions $Q_2, Q_3, \ldots, Q_T$ are defined recursively:

$$\begin{aligned} Q_2(x_1) = \min\{c_2x_2 + E_3Q_3(x_1,x_2) & \\ \text{subject to} \quad & x_2 \in \mathbb{R}^{n_2} \\ & \ell_2 \le x_2 \le u_2 \\ & A_{21}x_1 + A_{22}x_2 = b_2\}; \\ Q_3(x_1,x_2) = \min\{c_3x_3 + E_4Q_4(x_1,x_2,x_3) & \\ \text{subject to} \quad & x_3 \in \mathbb{R}^{n_3} \\ & \ell_3 \le x_3 \le u_3 \\ & A_{31}x_1 + A_{32}x_2 + A_{33}x_3 = b_3\}; \end{aligned}$$

and so forth, for $t = 4, \ldots, T-1$, where "E_t" represents expectation with respect to the random variables in period t, until finally

$$\begin{aligned} Q_T(x_1,\ldots,x_{T-1}) = \min\{c_Tx_T & \\ \text{subject to} \quad & x_T \in \mathbb{R}^{n_T} \\ & \ell_T \le x_T \le u_T \\ & A_{T1}x_1 + \cdots + A_{TT}x_T = b_T\}. \end{aligned}$$

The data defining this problem may be conveniently arranged in an LP formulation for a single realization of the random variables:

$$\text{(MP)} \qquad \begin{array}{ll} \text{objective:} & c_1x_1 + c_2x_2 + \ldots + c_Tx_T \\ \text{constraints:} & x_t \in \mathbb{R}^{n_t} \qquad t = 1,\ldots,T \\ & l_t \le x_t \le u_t, \quad t = 1,\ldots,T \\ & A_1x_1 \hphantom{+ A_{T2}x_2 + \ldots + A_{TT}x_T} = b_1 \in \mathbb{R}^{m_1} \\ & A_{21}x_1 + A_{22}x_2 \hphantom{+ \ldots + A_{TT}x_T} = b_2 \in \mathbb{R}^{m_2} \\ & \ldots \\ & A_{T1}x_1 + A_{T2}x_2 + \ldots + A_{TT}x_T = b_T \in \mathbb{R}^{m_T} \end{array}$$

All entries of the matrices A_{ts} and vectors: c_t, ℓ_t, u_t, b_t may be random (although in practice all but a few entries will be deterministic). The indices $t = 1, \ldots, T$ signify the periods of the problem; to each period t there corresponds a decision vector $x_t \in \mathbb{R}^{n_t}$. The lower block-triangular constraint system expresses the typical feature of these problems—the decisions of the prior periods constrain the decision of the current period explicitly (since those decisions are known) but the decision of the current period is affected only implicitly by the feasibility and costs of possible future (recourse) decisions.

The proposed format is most easily understood by considering the problem (MP) in stages of gradually increasing levels of detail. First we regard (MP) as a purely deterministic problem and create an input file following the MPSX convention, ignoring (non-random) zero entries. This file will identify an objective vector c, upper and lower bounds u and ℓ, a right hand side b, and a block lower-triangular matrix A, all expressed in the usual column-row format; we call this the *core file.*

Next we note the period structure of the problem, that certain decisions $x_t \in \mathbb{R}^{n_t}$ are made at times $t = 1, \ldots, T$. Here it is necessary only to specify which rows and columns from the core file correspond to which periods. It is most simply done by indicating the beginning column and row for each period t. This is done in the *time file.* Note that such a system relies on the proper sequencing of the core file—we require that the list of row names is in order from first period to last period and that the block lower-triangular matrix A has been entered in column order: first period columns first, and last period columns last.

Finally there remains the specification of the distributions of the random entries; this takes place in the *stoch file.* The simplest case occurs when all random entries are mutually *independent*—one merely indicates by keywords which entries are distributed as discrete, uniform, beta, gamma, etc., and supplies the appropriate parameters. One may even use the entries in the core file to supply some of the parameters. We consider two general types of dependency among the random entries: *blocks* and *scenarios.* A block is a random vector whose realizations are observed in a single, fixed period. A *scenario* is a more general type of stochastic structure which models dependencies across periods. Following Lane and Hutchinson [**11**], we visualize scenarios as paths in an *event tree.* Each path corresponds to a particular sequence of events through time, and is assigned a probability that is the product of the conditional probabilities of the separate events.

The organization of the data files is similar to the MPSX format [**6**; pp. 199–209]. Each data file contains a number of *sections*, some of which are optional. A *header line*, or *header*, marks the beginning of each section and may contain *keywords* to inform the user that the data to follow should be treated in a special way. Each header line is divided into two fields delineated by specific columns.

Header line

- columns 1 through 14: first word field
- columns 15 through 24: second word field

Most sections contain *data lines.* These are differently arranged for linear programs than for networks and each version will be described in the appropriate sections. Data lines are distinguished by a blank in the first column; the first word of a header line must, therefore, begin with a non-blank character. Finally, *comment lines* are indicated by an asterisk (*) in the first column and may appear anywhere.

We close this section with a comment to potential users. The header lines are a powerful device that permits the expression of a wide variety of problem types, however, one should not exploit it by tailoring a format convenient for one's own peculiar taste in test problems. That is definitely not in the spirit of our proposal! Rather we hope that this proposal offers sufficient flexibility to be useful and we trust that many problems will become available using the basic elements to the fullest extent—with tailored modifications only when absolutely necessary.

2. Standard Format for Linear Programs

2.1 Core File

The *core file* is sketched only briefly since it closely follows the MPSX standard [**6**]. The core file consists of sections introduced by header lines. Data lines follow the headers. We assume that all appropriate dimensioning has been done before the files are read (for example in a file similar to the SPECS file for the MINOS program [**12**]).

A data line is divided into six fields: three name fields, two numeric fields and one code field.

Data line for LP

- columns 2 and 3: code field
- columns 5–12: first name field
- columns 15–22: second name field
- columns 25–36: first numeric field
- columns 40–47: third name field
- columns 50–61: second numeric field

A *name* is treated as a character string and may contain any ASCII symbol. Only numbers with decimal point, or in scientific notation, may appear in numeric fields.

Core file sections for LP

1. `NAME` – starts the input file. The second word field is used to identify the problem.
2. `ROWS` section – each data line specifies the names of the objective c and rows of the matrix A in the first name field, and the type of constraint (`E, L, G, N`) in the code field. The list of row names must be in order from first period to last, preceded by the objective name(s).
3. `COLUMNS` section – each data line specifies the column names and the nonzero values of c and A. This must be done in column order.
4. `RHS` section – data lines specify nonzero entries of the righthand side b.
5. `BOUNDS` (optional) – data lines specify the bound, u and ℓ, using the codes `LO` or `UP` in the code field.
6. `RANGES` (optional) – cf. the MPSX standard [**6**].
7. `ENDATA` – informative. End of problem data.

Core file – example

```
NAME          problem name
ROWS
 N  OBJ
 L  ROW1
 E  ROW2
    ......
COLUMNS
    COL1      ROW1      value          ROW2      value
    COL1      ROW3      value          ROW4      value
              ......
    COL2      ROW1      value          ROW2      value
RHS           ......
    RHS       ROW1      value          ROW2      value
    ......
BOUNDS
 LO BND       COL1      value
    ......
RANGES
    ......
ENDATA
```

If an entry is *random* then the value field should contain a non-zero number (which may or may not be meaningful). This number *must not be omitted* from the core file. Integer variables may be indicated by using delimiters as described in the MPSX/MIP documentation [**7**] for those users who want to use mixed integer programming techniques.

2.2 Time File

The *time file* contains the information needed to specify the dynamic structure of the problem. It indicates which rows and columns (i.e., elements of the decision vector x) are to be identified with which period. The first line identifies it as a time file and gives a name to the problem. The next header line consists of a single word `PERIODS` in the first name field, and contains the keyword `LP` in the second name field to identify the problem as a pure LP. The first two name fields of the data lines identify the beginning row and column names for each period with the corresponding period name in the third name field.

Time file – example

```
TIME          problem name
PERIODS       LP
    COL1      ROW 1                    PERIOD1
    COL6      ROW 3                    PERIOD2
    COL8      ROW19                    PERIOD3
ENDATA
```

In this example: Columns COL1 through COL5 are PERIOD1 decision variables and COL6 and COL7 are PERIOD2 variables; rows ROW1 and ROW2 are PERIOD1 constraints and ROW3 through ROW18 are PERIOD2 constraints. All remaining rows and columns belong to PERIOD3.

Other possible keywords on the PERIODS line are NETWORK for pure network problems and MIXED for coupled LP/network problems. These are explained in some more detail in sections 3.2 and 4, respectively.

2.3 Stoch File

In the *stoch file* the distributions of the random variables are specified. As mentioned, we consider three varieties of distributions: *independent*, *blocks*, and *scenarios*. Each type will be treated in separate sections of this file; each section consists of a header line followed by data lines.

Stoch file – header lines

1. STOCH – informative. Identifies a new problem with a give name in the second word field.
2. INDEP section – specifies the distribution of all independent random entries in separate sections for each type.
3. BLOCKS section – specifies the joint distribution of all dependent random entries in separate sections for each type.
4. SCENARIOS section – specifies the scenarios.
5. ENDATA – informative. End of problem data.

2.3.1 A note on distributions

The purpose of the stoch file is to give the user the information needed to compute with the random variables. In many applications the distributions are discrete or discrete approximations of (absolutely) continuous distributions; thus the user needs, ultimately, to know what value the random variable takes and with what probability. The discrete case is straightforward—this information may be explicitly provided in the stoch file and then stored in appropriate data structures by the user. In the continuous case users may have their own discretization scheme and may need only to know the parameters and type of the continuously distributed random variables. Such data is easily provided; however, users must then process it themselves to obtain a discrete approximation. Finally there are cases where the random variables may be accessed only through a *user-supplied subroutine*—for example the output of a random number generator of nonstandard type. Alternatively, the user may be able to compute directly with certain continuous distributions and may build approximations to more general distributions based on them—for example the piecewise linear and piecewise quadratic distributions investigated by Wets [**17**] and Birge and Wets [**2**]. This information is easily transmitted using the various data structures described below.

2.3.2 Independent

Independent random variables are easily treated. We provide facilities for identifying entries that are distributed as discrete, uniform, beta, gamma, normal and log-normal; certainly other distributions may be considered. The `INDEP` header line is repeated for each new distribution type; entries with the same type are listed together following the appropriate header. The keyword in the second word field of the header identifies the distribution. The data lines indicate the entry by column and row in the first two name fields, and the distribution parameters are entered in the first and second numeric fields.

Discrete. For each discretely distributed entry one must specify the values and corresponding probabilities. The first two name fields identify the entry, and the first two numeric fields are the value field and probability field respectively. The intervening third name field contains the name of the period in which the random variable is realized (this information is ignored by the input routine but is useful to have made explicit for data consistency checking).

```
INDEP         DISCRETE
    COL1      ROW8          6.0        PERIOD2       0.5
    COL1      ROW8          8.0        PERIOD2       0.5
    RHS       ROW8          1.0        PERIOD2       0.1
    RHS       ROW8          2.0        PERIOD2       0.5
    RHS       ROW8          3.0        PERIOD2       0.4
```

In this example the entry `COL1/ROW8` takes value 6.0 with probability 0.5 and 8.0 with probability 0.5; and the righthand side of `ROW8` takes values $\{1.0, 2.0, 3.0\}$ with probabilities $\{0.1, 0.5, 0.4\}$ respectively. Of course the probabilities associated with an entry must total one.

Uniform. The endpoints of the interval are the only relevant parameters for uniformly distributed entries. These are entered into the first two numeric fields; the third name field is blank.

```
INDEP         UNIFORM
    COL1      ROW8          8.0        PERIOD2       9.0
```

In this example the random entry `COL1/ROW8` is uniformly distributed over the interval $[8.0, 9.0]$.

Normal. The normal distribution is specified by mean μ and variance σ^2 in the first two numeric fields.

```
INDEP         NORMAL
    COL1      ROW8          $\mu$          PERIOD2       $\sigma^2$
```

Beta, Gamma, Lognormal. The standard beta on $[0, 1]$, gamma on $[0, \infty)$, lognormal on $[0, \infty)$ are two-parameter families of distributions and may be handled in a similar fashion to the normal, using the standard descriptors as presented in, for example, Raiffa and Schlaifer [**16**]. An adjustment to other intervals could be effected within the framework of the linear transformations described below.

Subroutine. Some random entries may have distributions that are computed by subroutines, for example, empirical distributions which are discretely distributed but whose values may be randomly generated by user-supplied computer codes.

```
INDEP         SUB
    COL1      ROW8         blank       PERIOD2
```

This example indicates that the user must access a subroutine to generate an appropriate distribution for the entry COL1/ROW8.

2.3.3 Blocks

Blocks may be regarded as mutually independent random vectors. We provide for three distribution types: *discrete*, *subroutine*, or *linear transformation.* As in the independent case, blocks with common distribution types are grouped in the same section under a header line.

Discrete. The "values" of a block are actually vectors of values of the entries that make up the block, and to each value of a block there corresponds a probability. We need two sorts of data lines to describe a block. The first line, distinguished by a BL in the code field, gives the name of the block, the name of the period in which the block is realized, and the probability that the block assumes a given vector value; the following lines identify which entries of the block assume which value.

```
BLOCKS        DISCRETE
 BL BLOCK1    PERIOD2        0.5
    COL1      ROW6          83.0
    COL2      ROW8           1.2
 BL BLOCK1    PERIOD2        0.2
    COL2      ROW8           1.3
 BL BLOCK1    PERIOD2        0.3
    COL1      ROW6          84.0
```

One needs to record only those values that change. We adopt the convention that the first statement of the block is the basis from which all changes are computed. (**Thus zero values must be stated explicitly**.) In this example the block, called BLOCK1, is the 2-vector made up of the entries COL1/ROW6 and COL2/ROW8. It takes values $(83.0, 1.2)$ with probability 0.5, $(83.0, 1.3)$ with probability 0.2, and $(84.0, 1.2)$ with probability 0.3.

Subroutine. The user accesses a subroutine to compute the distribution of the block consisting of the listed entries.

```
BLOCKS         SUB
 BL BLOCK1     PERIOD2
    COL1       ROW6
    COL2       ROW8
 BL BLOCK2     PERIOD2
    RHS        ROW6
    RHS        ROW8
```

Here we have identified two blocks, each a 2-vector, BLOCK1 and BLOCK2 whose distributions must be computed by a subroutine.

Linear Transformations. These are blocks whose distribution is computed as a linear transformation of another random vector with independent components, i.e.,

$$v = Hu$$

where H is a matrix and v and u are random vectors. The vector v is the block whose distribution is desired; the vector u has independently distributed components of standard type, e.g., normal. We first identify the block as in the subroutine case, then the (marginal) distribution of each (independent) component of u followed immediately by the corresponding column of H.

```
BLOCKS         LINTR
 BL V_BLOCK    PERIOD2
    COL1       ROW8
    COL3       ROW6
 RV U1         NORMAL      μ           blank       σ^2
    COL1       ROW8        h_11
    COL3       ROW6        h_21
 RV U2         UNIFORM     a           blank       b
    COL1       ROW8        h_12
    COL3       ROW6        h_22
 RV U3         CONSTANT
    COL3       ROW6        C
```

This example illustrates the file structure. In this case

$$\texttt{COL1/ROW8} = h_{11} \cdot N(\mu, \sigma^2) + h_{12} \cdot U(a, b)$$
$$\texttt{COL3/ROW6} = h_{21} \cdot N(\mu, \sigma^2) + h_{22} \cdot U(a, b) + C.$$

General multivariate normal [**8**] and multigamma distributions [**13**] can also be treated in this way. Note that the "names" U1, U2 and U3 are irrelevant and may be left blank.

2.3.4 Scenarios

To describe *scenarios* one needs a data structure that expresses inter-period dependencies. This is best developed as a description of the distribution of a *process* vector in the periods, $t = 1, \ldots, T$, just as one may describe a stochastic process in probability theory. We consider the random entries of $(c_t, b_t, A_{ts} : s \leq t)$ as states of a process vector φ_t, for each time $t = 1, \ldots, T$. Given the corresponding (finite dimensional) joint distributions of this process φ, Kolmogorov's construction yields a probability measure P, termed the *process distribution*, on the space Ω of trajectories. Thus, in general, we have the alternatives of describing the distribution of the stochastic process φ in joint or conditional state distribution form, or as a process distribution over trajectories.

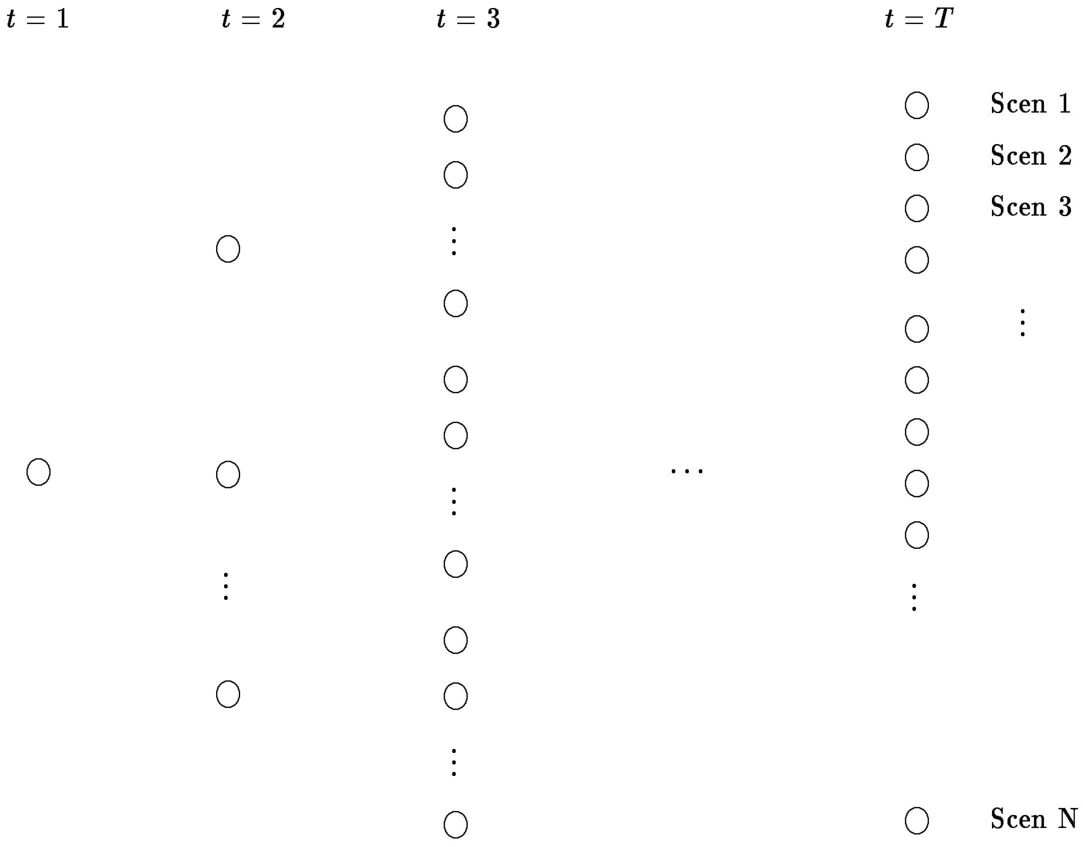

Figure 1. Event tree representation of scenarios

More specifically, with the stochastic process φ is associated a *filtration* $\{\mathcal{F}_t : t = 1, \ldots, T\}$, where for each t the sigma algebra $\mathcal{F}_t$ consists of subsets of Ω termed *events* determined by the *history* of the process φ up to time t, and $\mathcal{F}_t \subset \mathcal{F}_{t+1}$ for all $t = 0, \ldots, T-1$ (with $\mathcal{F}_0 := \{\Omega, \emptyset\}$). Given the process distribution P over the space of trajectories and an event A in period $t+1$ we may compute the *conditional probability* $P\{\varphi_{t+1} \in A \mid \mathcal{F}_t\}$. Conversely, these conditional probabilities may be composed to generate the finite dimensional distributions of the process φ. In the case under consideration in this paper, all this may be given a much simpler, more graphic, characterization.

To describe the process paths in the discrete state case, note that φ_t can assume only finitely many values for each t. A given history of values φ_s, for $s = 1, \ldots, t$, may be followed by a finite collection of values of φ_{t+1}. We think of these as *nodes* in period $t+1$. Following Lane and Hutchinson [**11**] and Raiffa [**15**], we construct an *event tree* representation of the trajectories: Represent the (unique) value of φ_1 by a single node connected by *oriented arcs* from φ_1 to nodes representing the values of φ_2. These are the *descendant* nodes of the first period node in the terminology of Birge [**1**]. Each node of period 2 is connected to its descendant nodes at period 3 by individual arcs oriented in the direction of the period 3 nodes. This construction is continued by connecting each *ancestor* at period $t-1$ with its descendants at period t. Each node of this tree has a single entering arc and multiple departing arcs representing the possible next period events.

A trajectory φ thus corresponds to a *path* from the period 1 node to a single period T node composed of arcs oriented in the direction of increasing time t and, moreover, corresponds uniquely to a single node in the last period T. There are only finitely many paths linking nodes in period 1 to nodes in period T and to specify the distribution on the paths one needs only to attach a definite probability to each path. Hence specifying the process distribution in terms of path probabilities can be effected in this context by assigning probabilities to period T nodes of the event tree (see Figure 2).

For each period $t = 1, \ldots, T$, the corresponding sigma algebra $\mathcal{F}_t$ is formed by taking all possible unions of the events represented by the nodes of period t; thus the topology of this event tree represents the information about the process expressed by the filtration $\{\mathcal{F}_t : t = 1, \ldots, T\}$. To each arc we attach the probability that the terminal node occurs given the initial node has occurred—that is, the conditional probability of φ_{t+1} given the history φ_s for $s = 1, \ldots, t$. These *arc probabilities* can be computed from the path probabilities by summing the probabilities of the paths visiting the terminal node and then dividing by the sum of the probabilities of the paths visiting the initial node. Conversely, if to each arc in the tree we know the conditional probability that its terminal node occurs given that its initial node has occurred, then the probability of any given path is simply the product of the arc probabilities along the path.

A decision x_t is made only on the basis of information collected up to and including time t. This is represented by a single node in period t. The uncertainty faced by the decision maker is represented by the collection of paths that branch from this node. Thus in Figure 2, the decisions occur at the nodes and the scenarios branch after the node.

In a language more specific to our application, a "path" in the tree analogy

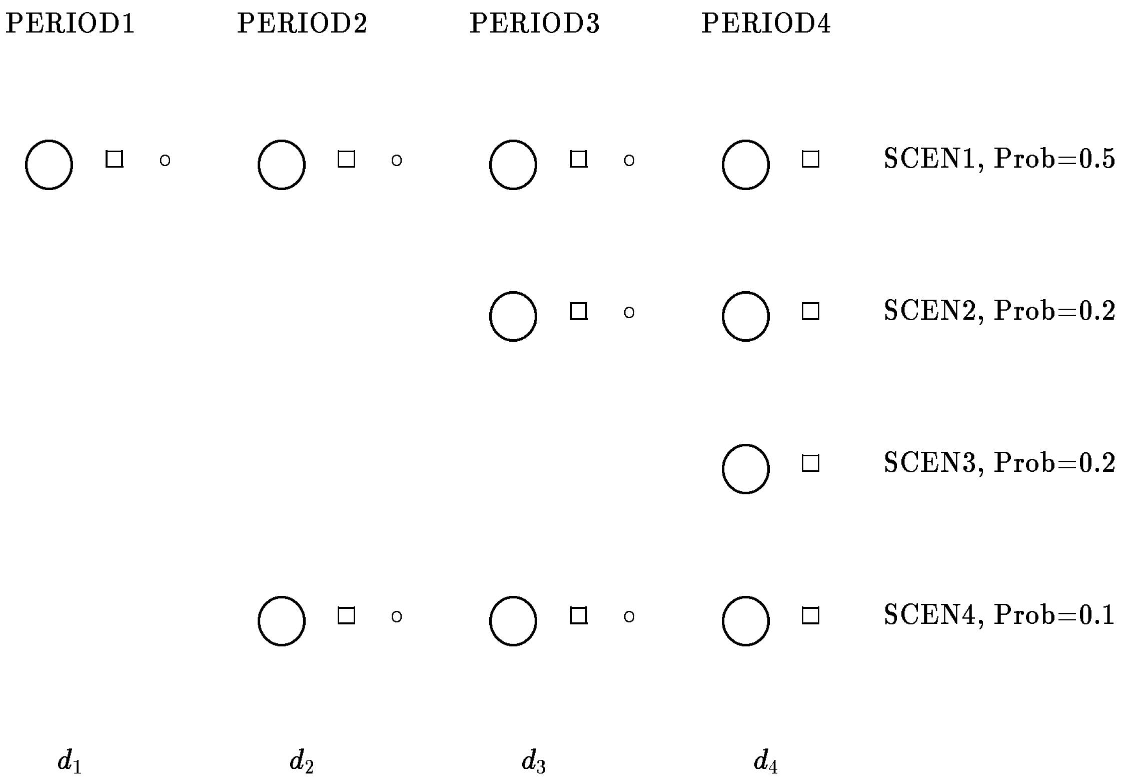

Figure 2. Scenarios example – event tree

is a single "scenario". The nodes visited by the path correspond to certain values assumed by certain entries of the matrices in the core file. Thus a scenario is completely specified by a list of column/row names and values, and a probability value. Once a single given scenario is described, then other scenarios that branch from it may be described by indicating in which period the branch has occurred, and then listing the subsequent column/row names and values. It is best to work through the example of Figure 2.

There are two types of data lines. The first, signified by `SC` in the code field, gives the name of the scenario in the first name field and its probability in the first numeric field; and then gives the name of the scenario from which the branch occurred and the name of the period in which the branch occurred—i.e., the first period in which the two scenarios *differ*—in the second name field and third name field, respectively. A scenario that originates in period one is indicated by `ROOT` in the name field. The next data lines give the column/row values assumed by the scenario.

Scenarios – example

```
SCENARIOS     DISCRETE
 SC SCEN1     ROOT          0.5         PERIOD1
    COL1      ROW2          1.0
    COL2      ROW3          1.0
    COL3      ROW4          1.0
    COL4      ROW5          1.0
 SC SCEN2     SCEN1         0.2         PERIOD3
    COL3      ROW4          1.0
    COL4      ROW5          1.0
 SC SCEN3     SCEN2         0.2         PERIOD4
    COL4      ROW5          0.0
 SC SCEN4     SCEN1         0.1         PERIOD2
    COL2      ROW3          0.0
    COL3      ROW4          0.0
    COL4      ROW5          0.0
```

This is a description of the distribution of four entries: `COL1/ROW2`, `COL2/ROW3`, `COL3/ROW4`, `COL4/ROW5`, which for convenience we denote here as d_1, d_2, d_3, d_4, respectively (see Figure 2). Note that in `PERIOD4` there are two nodes for the "state" $d_4 = 0.0$ and two for $d_4 = 1.0$, and similarly in `PERIOD3` two nodes for $d_3 = 1.0$. This is because a node is distinguished by the information that one has collected concerning the path up to and including time t. Thus in `PERIOD3` the two nodes are distinguished because in scenario `SCEN1` one *knows* that the final state is $d_4 = 1.0$, whereas in `SCEN2` the outcome of d_4 is in doubt.

3. Network Standard Format

There are several network formats in use. However, it is our view that the most common format is the NETGEN format, see Klingman, Napier and Stutz [**10**]. For NETGEN we have the following organization of the data line.

Data line for networks

- columns 2–4: code field
- columns 7–12: first name field
- columns 13–18: second name field
- columns 21–30: first numeric field
- columns 31–40: second numeric field
- columns 41–50: third numeric field
- columns 51–60: fourth numeric field

3.1 Core File

As in MPSX there are header lines and data lines in the input format. We adopt a slight variation of the NETGEN standard in omitting the BEGIN line and substituting a `NAME` line to start the input file and in changing the END line to `ENDATA`.

Core file sections for networks

1. NAME – starts the input file. The rest of the line can be used for the problem name.
2. ARCS section – each data line following the ARCS header specifies input for one arc. In the first name field is the name of the *originating node* for the arc, in the second name field the name of the *terminating node*, in the first numeric field the *unit cost*, in the second numeric field the *upper bound* on arc flow, in the third numeric field the *lower bound* on the flow (if not zero) and in the fourth numeric field the *arc multiplier* (arc gain) if we are dealing with generalized networks. If the word UNCAP follows the ARCS code, upper and lower bounds need not be specified as they are assumed to be ∞ and zero, respectively. Similarly, the keyword UNDIR signals an undirected network with default bounds of $+\infty$ and $-\infty$.
3. SUPPLY (optional) – each data line following the SUPPLY header contains a node name in the first name field and the amount supplied in the second numeric field.
4. DEMAND (optional) – each data line following the DEMAND header contains a node name in the first name field and the amount demanded in the second numeric field.
5. ENDATA – informative. End of problem data.

Note that there is no section naming all nodes (i.e., rows). They are named implicitly by their appearance in the ARCS section. Also note that arcs (i.e., columns) have no names. Hence they cannot be referred to by name, only by a pair of node names. However, if there are parallel arcs, the user must be careful.

Core file – example

```
NAME        problem name
ARCS
    NODE1   NODE2   cost      upper     lower     multiplier
    NODE1   NODE3   cost      upper     lower     multiplier
    ......
    NODEk   NODEn   cost      upper     lower     multiplier
SUPPLY
    ......
    NODE1                     amount
DEMAND
    ......
ENDATA
```

3.2 Time File

It is normally assumed that in a NETGEN file, all arcs originating in a given node are given before we start giving arcs originating in the next node. This rule should be followed. We shall further assume that FROM-nodes are given in node order, in the sense that if the arcs originating in NODEi occur before those originating in NODEj, then NODEj belongs to the same or a later time period. However, note that we are not able to give arcs in arc order at the same time. The keyword NETWORK in the PERIODS header indicates that the problem is a pure network.

Time file – example

```
TIME            problem name
PERIODS         NETWORK
                NODE1               PERIOD1
                NODE3               PERIOD2
                NODE7               PERIOD3
ENDATA
```

In the example, NODE1 and NODE2 are first period nodes, NODE3 to NODE6 are second period nodes, and all remaining nodes are third period nodes. Arcs going from first to second period nodes are a part of the first period decisions, and carry flow over to the second period by defining external flows for that period.

3.3 Stoch File

Since arcs are defined by *pairs* of nodes, to say that the cost of the arc from node NODE1 to node NODE2 is random requires three parameters. The same goes for bounds and multipliers. We can use the CODE field for that purpose as in the following example. (The data line format used in the stoch file for the networks case is the same as that for the LP case.)

```
INDEP           DISCRETE                    NETWORK
 C1 NODE1       NODE3           6.0         PERIOD2         0.6
 C1 NODE1       NODE3           8.0         PERIOD2         0.4
 U2 NODE6       NODE8           7.0         PERIOD2         0.2
 U2 NODE6       NODE8           9.0         PERIOD2         0.8
 SU NODE6                       6.0         PERIOD2         0.1
 SU NODE6                       8.0         PERIOD2         0.9
```

The keyword NETWORK indicates that there is something to look for in the code field of the subsequent data records. We use C1 for “cost of first arc from node NODE1 to node NODE2” and use higher numbers, such as C2, for the second parallel arc, etc. Similarly, M = multiplier, U = upper bound, and L = lower bound. Random supply and demand are accomodated by use of SU and DE, respectively, in the code fields as illustrated. This format can be changed to all the other distribution forms in obvious ways.

4. Coupled LP and Network Formats

A network can be viewed as a special case of a linear program, yet it is desirable to use the more compact network data format to express those parts of a multistage problem that may be interpreted as network flow problems. The nodes of a network flow problem are the rows in its LP statement. Our proposal for the coupled format utilizes an MPSX-like format, but uses the ARCS card to indicate that the following data lines are in the NETGEN format and the COLUMNS card to indicate that the following lines are in MPSX format. While this is consistent with the philosophy as expressed in the introduction, it does have a particular disadvantage. Many multistage mixed LP/network problems will actually be composed from separate problem files. To generate the proposed format will involve a certain amount of editing of these files, however this editing could be automated in various (system specific) ways.

The first section is the ROWS. (The absence of a ROWS card tells the program that the problem to follow is a pure network.) Note that only those node names that are going to appear as row names of an LP variable need to be named in the ROWS section. Then follows the COLUMNS/ARCS section. Once the data for the columns and arcs has been entered, then the other sections (RHS, BOUNDS, SUPPLY, etc.) may follow. With the proper logic, the same computer program can read all three formats and does not need to be told beforehand the nature of the problem in the file, whether pure LP, pure network, or mixed. As always, the end of the problem is indicated by an ENDATA line. Below is an example for the case when we first have LP, then networks and then LP again.

Core file – example of coupled format

```
NAME          problem name
ROWS
    ......
 E  ROW3
 E  ROW4
 L  NODE1
 G  NODE2
 E  ROW5
 E  ROW6
    ......
COLUMNS
    ......
    COL16     ROW3      value          NODE1     value
    COL17     ROW4      value          NODE2     value
ARCS
    NODE1   NODE2   cost      upper     lower     multiplier
    NODE1   NODE7   cost      upper     lower     multiplier
    ......
    NODE8   ROW5    cost      upper     lower     multiplier
    NODE8   ROW6    cost      upper     lower     multiplier
COLUMNS
    COL18     ROW5      value          ROW7      value
    COL19     ROW6      value          ROW8      value
    ......
RHS
    ......
SUPPLY
    ......
ENDATA
```

If we now have a TIME file as follows, the coupling is done.

Time file

```
TIME          problem name
PERIODS       MIXED
    COL1      ROW1                PERIOD1
    COL6      ROW3                PERIOD2
              NODE1               PERIOD3
              NODE3               PERIOD4
    COL18     ROW5                PERIOD5
ENDATA
```

That the problem is a mixed LP/network is indicated by the keyword `MIXED`. Here we have that `COL1` through `COL5` are first stage decision variables, with `ROW1` and `ROW2` first stage constraints. `COL6` through `COL17` are second stage decision variables, with `ROW3` and `ROW4` as constraints. Arcs originating in `NODE1` and `NODE2` are third stage decision variables, arcs originating in nodes `NODE3` through `NODE8` belong to stage 4. Finally, all variables corresponding to columns `COL18` through whatever the second `COLUMNS` section dictates are fifth stage decisions with all constraints after `ROW5` associated with them.

The last two entries in the first `COLUMNS` section will take values from the second to the third stage. The amount will be determined by the values of `COL16` and `COL17` and the corresponding entries in this `COLUMNS` section.

The last two entries in the `ARCS` section will bring "flow" from `NODE8` to the right hand side of `ROW5` and `ROW6` by entering a number in those rows. The number will be the "multiplier" from the input, and the value ending up on the right hand side will be the product of this multiplier and the flow running out of `NODE8` to these rows (which are nodes when viewed from the network).

Acknowledgements

The authors are grateful to Dalhousie University and the Natural Sciences and Engineering Research Council of Canada for providing facilities and support for this research.

References

[1] J.R. Birge, "Decomposition and partitioning methods for multistage stochastic linear programs", *Operations Research* **33**(1985), 989–1007.

[2] J.R. Birge and R. J-B Wets, "A sublinear approximation method for stochastic programming", Department of Industrial and Operations Engineering, The University of Michigan, Technical Report 86-26.

[3] G. Dantzig, *Linear Programming and Extensions*, Princeton University Press, 1968.

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

[5] Yu. Ermoliev and R.J-B. Wets, *Numerical Methods for Stochastic Programming*, Springer Verlag, 1987.

[6] International Business Machines, Inc., *Mathematical Programming Subsystem — Extended (MPSX) and Generalized Upper Bounding (GUB) Program Description*, document number SH20-0968-1, 1972.

[7] International Business Machines, Inc., *Mathematical Programming Subsystem — Extended (MPSX) and Mixed Integer Programming (MIP) Program Description*, document number GH19-1091-0.

[8] N.L. Johnson and S. Kotz, *Distributions in Statistics*, Vol. 4, Wiley, New York, 1972.

[9] A.J. King, "Stochastic programming problems: examples from the literature", in Yu. Ermoliev and R.J-B. Wets, *op cit.*

[10] 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.

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

[12] B.A. Murtagh and M.A. Saunders, "MINOS – User's Guide", Technical Report SOL 77-9, Systems Optimization Laboratory, Department of O.R., Stanford Univ.

[13] A. Prékopa and T. Szántai, "A new multivariate gamma distribution and its fitting to empirical data", *Water Resources Research* 14 (1978), 19–24.

[14] A. Prékopa and R.J-B Wets, *Stochastic Programming: 1984*, Math. Prog. Study **27**, 1986.

[15] H. Raiffa, *Decision Analysis: Introductory Lectures on Choices Under Uncertainty*, Addison-Wesley, Reading, 1968.

[16] H. Raiffa and R. Schlaifer, *Applied Statistical Decision Theory*, Harvard University Press, Cambridge, 1961.

[17] R.J-B Wets, "Solving stochastic programs with simple recourse, II", in: Proceedings of 1975 Conference on Information Sciences and Systems, Johns Hopkins Univ. Press, Baltimore, Maryland, 1975.